View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by DukeSpace

Security Games: Solution Concepts and Algorithms

by

Dmytro Korzhyk

Department of Computer Science
Duke University

Date: _____

Approved:

_____
Vincent Conitzer, Supervisor

_____
Ronald Parr

_____
Kamesh Munagala

_____
Milind Tambe

Dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in the Department of Computer Science
in the Graduate School of Duke University
2013

Abstract

Security Games: Solution Concepts and Algorithms

by

Dmytro Korzhyk

Department of Computer Science
Duke University

Date: _____

Approved:

_____
Vincent Conitzer, Supervisor

_____
Ronald Parr

_____
Kamesh Munagala

_____
Milind Tambe

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science in the Graduate School of Duke University
2013

Copyright © 2013 by Dmytro Korzhyk
All rights reserved

Abstract

Algorithms for finding game-theoretic solutions are now used in several real-world security applications. These applications are based on different but related game-theoretical models collectively known as security games. Much of the research in this area has focused on the two-player setting in which the first player (leader, defender) commits to a strategy, after which the second player (follower, attacker) observes that strategy and responds to it. This is commonly known as the Stackelberg, or leader-follower, model. In contrast, if none of the players can observe the actions of the others then such a setting is called a simultaneous-move game. A common solution concept in simultaneous-move games is that of Nash equilibrium (NE). In the present dissertation, we contribute to this line of research in two ways.

First, we consider new ways of modeling commitment. We propose a new model in which the leader can commit to a correlated strategy. We show that this model is equivalent to the Stackelberg model in two-player games and is different from the existing models in games with three or more players. We propose an algorithm for computing a solution in this model in polynomial time. We also consider a leader-follower setting in which the players are uncertain about whether the follower can observe the leader's strategy. We describe an iterative algorithm for solving such games.

Second, we analyze the computational complexity of computing Stackelberg and NE strategies in security games. We describe algorithms to solve some variants of

a previously proposed model of security games in polynomial time and prove NP-hardness of solving other variants of the model. We also extend the family of security games by allowing the attacker to have multiple resources. We provide an algorithm for computing an NE of such games in polynomial time, and we show that computing a Stackelberg strategy is NP-hard.

Contents

List of Tables

List of Figures

List of Abbreviations and Symbols

General game-theoretic notation

The symbols in the following list are used to describe the various games throughout the thesis.

A_i	The set of actions of player i. We will denote the individual actions by, for example, $a_i, a_i' \in A_i$.
S_i	The set of mixed strategies of player i. Each $s_i \in S_i$ is a distribution over A_i.
u_i	The utility function of player i. For example, in two-player games, player i's utility for the outcome (a_1, a_2) is $u_i(a_1, a_2)$.
$u_i(s_1, s_2)$	The expected utility of player i when the players play the mixed strategies s_1, s_2.

Security games notation

The symbols in the following list are used to describe the security games models in Chapters 4 and 5.

T	The set of targets. We will usually refer to the individual targets using symbols such as t, t', t_1, t_2.
Ω	The set of resources.
Σ	The set of schedules. Each schedule $\sigma \in \Sigma$ is a subset of targets T.
H	When the resources are heterogeneous, we denote the subset of schedules to which resource ω can be assigned by $H(\omega)$.

$u_d^c(t), u_d^u(t)$	The utility that the defender receives if target t is attacked and is covered by a resource (superscript c) / not covered by a resource (superscript u).
$u_a^c(t), u_a^u(t)$	Similar to the above but for the attacker's utility.
c_t	The probability that target t is covered by a resource.
$c_{\omega,\sigma}$	The probability that resource ω is assigned to schedule σ.

Acknowledgements

I would like to thank my wife Jun and my parents for their support during the time of my studies at Duke. I also thank everybody with whom I collaborated on research projects while at Duke, including Sayan Bhattacharya, Vincent Conitzer, Melissa Dalis, Charles Isbell, Manish Jain, Christopher Kiekintveld, Joshua Letchford, Liam MacDermed, Kamesh Munagala, Ronald Parr, Michal Pěchouček, Troels Sørensen, Milind Tambe, Ondřej Vaněk, Zhengyu Yin. I would also like to thank the participants of Duke CS-Econ seminar series, Duke Robotics Intelligence and Vision seminar, Duke Algorithms seminar, and Duke Microeconomic Theory seminar, the participants of the conferences in which I presented my work, and the reviewers of my papers for their comments.

I would also like to acknowledge NSF CAREER 0953756 and IIS-0812113, ARO Grant W911NF-09-1-0459, DARPA CSSG HR0011-06-1-0027, and a J.B. Duke fellowship for support.

1

Introduction

When multiple self-interested agents interact in the same domain, *game theory* provides a framework for reasoning about how each agent should act. One use of game theory is by an outside party that tries to predict the outcome of a strategic situation. For example, when we design a mechanism (e.g., an auction), we can use game theory to evaluate whether any given design will lead to good outcomes when the agents participating in it are strategic. Another use is by one of the agents in the game that wants to determine how to play. For example, game theory is often used to create poker-playing programs (Sandholm, 2010). Recently, algorithms for computing game-theoretic solutions have also started to find applications in security applications, where one of the players, the defender, tries to allocate limited defensive resources in anticipation of an attack by an attacker. Real-world examples include the placement of checkpoints and canine units at Los Angeles International airport (Pita et al., 2009), the assignment of Federal Air Marshals to flights (Tsai et al., 2009), scheduling of Coast Guard patrols (An et al., 2013), and scheduling patrols of the Los Angeles Metro rail to deter fare evasion (Jiang et al., 2013).

1.1 Nash Equilibrium

Probably the best-known solution concept in game theory is that of *Nash equilibrium* (NE, Nash 1950). Before giving a formal definition, we will first show a Nash equilibrium of the example game in Figure 1.1 and introduce some of the notation that we will use throughout this thesis.

	L	R
U	(1,1)	(3,0)
D	(0,0)	(2,1)

FIGURE 1.1: An example 2×2 normal-form game.

There are two players in this game. The row player's set of actions (or *pure strategies*) is $A_1 = \{U, D\}$. The column player's set of actions is $A_2 = \{L, R\}$. Depending on the players' choice of actions, there are four possible *outcomes* of the game, each specified by a pair of actions (a_1, a_2). The table representation shown in Figure 1.1 is called the *normal form* of the game. In the normal form, the utilities for each outcome are specified by an entry in the matrix. In this example, if the outcome of the game is $(a_1 = U, a_2 = R)$, then the row player gets a utility of $u_1(U, R) = 3$, and the column player gets $u_2(U, R) = 0$.

Any game with a finite number of pure strategies for each player has a normal form representation. In this thesis, we will use the normal form as well as other game representations.

If the two players in the example are rational, we can compute the unique Nash equilibrium using *iterated strict dominance* as follows. First, note that the row player gets a higher utility from playing U than she would get from playing D no matter which action the column player chooses. We say that the action U *dominates* the action D for the row player, and we can thus remove the action D from the game.

Once the D row is removed from the game, the L action dominates the R action for the column player, and we can remove the R column from the game. The only outcome left, (U, L), is the unique *pure-strategy Nash equilibrium* of this game.

A pure-strategy Nash equilibrium is a *profile* of actions, one for each player, such that no player can increase her utility by unilaterally changing her action. Not every game has a pure-strategy Nash equilibrium. However, every normal-form game has a Nash equilibrium in mixed strategies (Nash, 1950). A *mixed strategy* is a probability distribution over a player's action set. A *mixed-strategy Nash equilibrium* is a profile of mixed strategies such that no player can increase her expected utility by unilaterally changing her mixed strategy. In this thesis, when we talk about a strategy, we will usually mean a mixed strategy, and when we talk about a Nash equilibrium, we will mean a mixed-strategy Nash equilibrium, unless otherwise noted.

How can the players achieve a Nash equilibrium in a game? If the equilibrium is unique, then each player can compute the equilibrium profile and play her strategy in that profile. In some games with multiple equilibria, the players can always achieve an equilibrium by each computing an arbitrary equilibrium profile and playing her part of that profile, even if the players do not coordinate on the same equilibrium. This property is called the *interchange property*. If the interchange property does not hold, the players may not be able to achieve an equilibrium without using extra communication: if they compute different equilibria, the result may not be an NE. In such games, the players face the *equilibrium selection problem*.

Computing a Nash equilibrium of a normal-form game is PPAD-complete (Daskalakis et al., 2009; Chen et al., 2009), and computing an (even approximately) optimal Nash equilibrium is NP-hard for just about any reasonable definition of optimality (Gilboa and Zemel, 1989; Conitzer and Sandholm, 2008).

Even though there is no known polynomial-time algorithm for computing a Nash equilibrium of normal-form games, it may still be possible to find a polynomial-time

algorithm for computing an NE of games which have a special structure in the utility function. On the other hand, for games which have more concise representations than the normal form, it may be possible to prove NP-hardness in the size of the concise representation, even when just computing any one equilibrium. We will consider several games with concise representations (in Chapters 4 and 5) and provide polynomial-time algorithms for computing an NE of some of those games and prove NP-hardness of computing an NE in other games. We will also prove the interchange property in certain games. But before we get to that, we will discuss another solution concept, the (mixed) Stackelberg strategy, and the relationship between the NE and the Stackelberg strategy.

1.2 The Stackelberg Model

Nash equilibrium is not the only solution concept in game theory. An alternative solution concept (for two-player games) is the following. Suppose that player one (the *leader*) is able to *commit* to a mixed strategy; then, player two (the *follower*) observes this commitment, and chooses a response. Such a commitment model is known as a *Stackelberg* model (von Stackelberg, 1934), and we will refer to an optimal mixed strategy for player one to commit to as an (optimal) Stackelberg strategy.

It has long been well known in game theory that being able to commit to a course of action before the other player(s) move(s)—often referred to as a *Stackelberg* model (von Stackelberg, 1934)—can bestow significant advantages. Consider again the example game in Figure 1.1. If player 1 can commit to a pure strategy before player 2 moves, then player 1 is better off committing to D, thereby incentivizing player 2 to play R, resulting in a utility of 2 for player 1. Even better for player 1 is to commit to a mixed strategy of $(.49U, .51D)$; this still incentivizes player 2 to play R and results in an expected utility of $.49 \cdot 3 + .51 \cdot 2 = 2.49$ for player 1. Of course, it is even better to commit to $(.499U, .501D)$, etc. In the limit case of $(.5U, .5D)$, player

2 becomes indifferent between L and R; to guarantee the existence of an optimal solution, it is generally assumed that player 2 breaks ties in player 1's favor, so that $(.5U, .5D)$ is the unique optimal mixed strategy for player 1 to commit to, resulting in an expected utility of 2.5 for her. In two-player *zero-sum* games, Nash equilibrium strategies and Stackelberg strategies both coincide with minimax strategies (and, hence, with each other), due to von Neumann's minimax theorem (von Neumann, 1928).

In recent years, the problem of *computing* an optimal strategy to commit to in non-zero-sum games has started to receive a significant amount of attention, especially in the multiagent systems community. In the initial paper (Conitzer and Sandholm, 2006), a number of variants were studied, including commitment to pure and to mixed strategies, in normal-form and in Bayesian games. There have been several other papers making progress on versions of the problem that concern standard game-theoretic representations, including Bayesian games (Paruchuri et al., 2008; Letchford et al., 2009; Jain et al., 2011), extensive-form games (Letchford and Conitzer, 2010), and stochastic games (Letchford et al., 2012). Perhaps the biggest impulse to this line of research is due to the use of these techniques in several security applications, which we mentioned earlier in this chapter. These developments have inspired work on computing optimal mixed strategies to commit to in a specific class of games called *security games* (Kiekintveld et al., 2009; Korzhyk et al., 2010).

We will first describe the standard algorithm for computing a Stackelberg strategy in a normal-form game. We will use some of the ideas from this algorithm to design new algorithms in this thesis.

1.3 The Multiple-LPs Approach to Computing a Stackelberg Strategy

For a two-player normal-form game (not necessarily zero-sum), the optimal Stackelberg strategy can be found in polynomial time, using a set of linear programs

(LP) (Conitzer and Sandholm, 2006; von Stengel and Zamir, 2010).[1] Besides this computational benefit over Nash equilibrium, with Stackelberg strategies there is effectively no equilibrium selection problem.

We now describe the standard approach to computing an optimal mixed strategy to commit to with two players. The idea is a very natural divide-and-conquer approach: because we can assume without loss of optimality that in the solution, player 2 will play a pure strategy, we can simply consider each pure strategy for player 2 in turn. Let player i's set of pure strategies be A_i. For each pure strategy $a_2 \in A_2$ for player 2, we solve for the optimal mixed strategy for player 1, *under the constraint that a_2 is a best response for player* 2.

Linear Program 1 (known).

$\max \sum_{a_1 \in A_1} p_{a_1} u_1(a_1, a_2)$

subject to:

$(\forall a_2' \in A_2) \ \sum_{a_1 \in A_1} p_{a_1} u_2(a_1, a_2') \leqslant \sum_{a_1 \in A_1} p_{a_1} u_2(a_1, a_2)$

$\sum_{a_1 \in A_1} p_{a_1} = 1$

$(\forall a_1 \in A_1) \ p_{a_1} \geqslant 0$

(The first constraint says that player 2 should not be better off playing a_2' instead of a_2.) There is one of these linear programs for every a_2, and at least one of these must have a feasible solution. We choose one with the highest optimal solution value; an optimal solution to this linear program corresponds to an optimal mixed strategy to commit to. Because linear programs can be solved in polynomial time, this gives a polynomial-time algorithm for computing an optimal mixed strategy to commit to.

[1] It is not known whether linear programs are solvable in *strongly* polynomial time, that is, with no dependence on the sizes of the input numbers at all. Consequently, it is not known whether any of the problems for which we propose LP-based solutions in this thesis can be solved in strongly polynomial time.

We will now discuss several computational and modeling issues with the Stackelberg model and briefly describe how we address those issues in this thesis.

It is not clear how best to extend the Stackelberg model to games with three or more players. Should there be a single leader who commits to a strategy and lets the other players play a Nash equilibrium in the resulting subgame? In such a model, the remaining players would face the equilibrium selection problem, and they would not have a polynomial-time algorithm to compute their strategies. Should we then let all the players commit in sequence? In that case, it may be not clear how to choose the right sequence of players, especially if the number of players is large. In Chapter 2, we propose another model, in which a single leader chooses a correlated strategy to commit to. We will show that our proposed model coincides with the Stackelberg model for games with two players and can be naturally extended to games with three or more players. We also provide an algorithm for solving our model in polynomial time.

In some practical applications, the players may be uncertain about whether the follower can indeed observe the leader's actions. Neither the Nash nor the Stackelberg model can be directly applied to such games. In Chapter 3, we consider one way of modeling such uncertainty about observability. We design an algorithm for computing an equilibrium of that model and provide experimental analysis of its run time.

When a game has a concise representation such that the size of its normal form is exponential in the size of the concise representation, the multiple-LPs approach described above is inefficient. We will consider several such games in Chapters 4 and 5. We will provide polynomial-time algorithms for computing a Stackelberg strategy in some of those games and prove that it is NP-hard to do in others.

1.4 The Dissertation Structure

In Chapter 2, we propose a new solution concept in which one of the players chooses a correlated strategy to commit to. We show the relationship between this new concept and Stackelberg strategies, and provide an algorithm for efficient computation of an optimal correlated strategy to commit to. In Chapter 3, we consider a game model in which the players are uncertain about whether the follower can observe the leader's strategy. We provide an algorithm for computing an NE in this extended game model and demonstrate practical limitations of such a model. In Chapter 4, we shift our attention to a class of games between a defender and an attacker known as security games (Kiekintveld et al., 2009). In security games, the defender first allocates resources to protect a set of targets, after which the attacker chooses a single target to attack. We discuss why the known techniques for solving normal-form games are inefficient when applied to security games and provide algorithms for efficiently solving security games. Finally, in Chapter 5, we expand the security games model by allowing simultaneous attacks. We discuss potential practical applications of the proposed model, provide an algorithm for computing an NE of the expanded game, and prove that computing a Stackelberg strategy is NP-hard.

2

Commitment to Correlated Strategies

In the introduction, we showed that in a two-player normal-form game, the optimal mixed strategy to commit to can be found in polynomial time, by solving multiple linear programs. Still, given the problem's fundamental nature and importance, it seems worthwhile to investigate it in more detail. Can we design other, possibly more efficient algorithms? Can we relate this problem to other computational problems in game theory?

With three or more players, it is not immediately clear how to define the optimal mixed strategy to commit to for player 1. The reason is that, after player 1's commitment, the remaining players play a game among themselves, and we need to consider according to which solution concept they will play. For example, to be consistent with the tie-breaking assumption in the two-player case, we could assume that the remaining players will play according to the Nash equilibrium of the remaining game that is best for player 1. However, this optimization problem is already NP-hard by itself, and in fact inapproximable unless P=NP (Gilboa and Zemel, 1989; Conitzer and Sandholm, 2008), so there is little hope that this approach will lead to an efficient algorithm. Is there another natural solution concept that allows for a more

efficient solution with three or more players?

In this chapter, we make progress on these questions as follows. First, we show how to formulate the problem in the two-player setting as a single linear program,[1] and prove the correctness of this formulation by relating it to the existing multiple-LPs formulation. We then show how this single LP can be interpreted as a formulation for finding the optimal *correlated* strategy to commit to, giving an easy proof of a known result by von Stengel and Zamir (2010) that the optimal mixed strategy to commit to results in a utility for the leader that is at least as good as what she would obtain in any correlated equilibrium. We then show how this formulation can be extended to compute an optimal correlated strategy to commit to with three or more players, and illustrate by example that this can result in a higher utility for player 1 both compared to the best mixed strategy to commit to as well as compared to the best correlated equilibrium for player 1. (Unlike in two-player games, in games of three or more players, the notions of optimal mixed and correlated strategies to commit to are truly distinct.) Finally, we present experiments that indicate that, for 50×50 games drawn from "most" distribution families, our formulation is significantly faster than the multiple-LPs approach. We also investigate how often the correlated strategy is a product distribution (so that correlation does not play a role); as expected, with two players we always have a product distribution, but with more players this depends extremely strongly on the distribution over games that is used.

[1] Earlier work has already produced formulations of ((pre-)Bayesian versions of) the problem that involve only a single optimization (Paruchuri et al. 2008, extended version of Letchford, Conitzer, and Munagala 2009). However, these existing formulations use integer variables (even when restricted to the case of a single follower type).

2.1 Review: Correlated Equilibrium and the LP to Compute It

In a *correlated equilibrium* (Aumann, 1974), a third party known as a *mediator* draws an outcome (a_1, a_2), recommends to each player i to play a_i without telling what the recommendation to the other player is, and it is optimal for each player to follow the recommendation. Every Nash equilibrium is also a correlated equilibrium. We can describe the set of all correlated equilibria of a normal-form game with a set of linear constraints. Consider the following linear feasibility formulation of the correlated equilibrium problem.

Linear Program 2 (known).

(no objective required)

subject to:

$(\forall a_1, a_1' \in A_1) \sum_{a_2 \in A_2} p_{(a_1,a_2)} u_1(a_1', a_2) \leqslant \sum_{a_2 \in A_2} p_{(a_1,a_2)} u_1(a_1, a_2)$

$(\forall a_2, a_2' \in A_2) \sum_{a_1 \in A_1} p_{(a_1,a_2)} u_2(a_1, a_2') \leqslant \sum_{a_1 \in A_1} p_{(a_1,a_2)} u_2(a_1, a_2)$

$\sum_{a_1 \in A_1, a_2 \in A_2} p_{(a_1,a_2)} = 1$

$(\forall a_1 \in A_1, a_2 \in A_2)\ p_{(a_1,a_2)} \geqslant 0$

The variables $p_{(a_1,a_2)}$ are the probabilities that the mediator puts on each outcome. The first set of constraints are the optimality constraints for player 1: for any recommendation that player 1 may receive, following that recommendation must give player 1 an expected utility at least as high as the expected utility from taking any other action. The second set of constraints similarly describes optimality for player 2.

In the next section, we will show how a similar LP can be used to compute a Stackelberg strategy.

2.2 A Single LP to Compute a Stackelberg Strategy

Instead of solving one separate linear program per pure strategy for player 2 as in the multiple-LPs approach, we can solve the following single linear program:

Linear Program 3.

$$\max \sum_{a_1 \in A_1, a_2 \in A_2} p_{(a_1,a_2)} u_1(a_1, a_2)$$

subject to:

$$(\forall a_2, a_2' \in A_2)$$

$$\sum_{a_1 \in A_1} p_{(a_1,a_2)} u_2(a_1, a_2') \leqslant \sum_{a_1 \in A_1} p_{(a_1,a_2)} u_2(a_1, a_2)$$

$$\sum_{a_1 \in A_1, a_2 \in A_2} p_{(a_1,a_2)} = 1$$

$$(\forall a_1 \in A_1, a_2 \in A_2)\ p_{(a_1,a_2)} \geqslant 0$$

We now explain why this linear program computes a Stackelberg strategy. The constraint matrix for Linear Program 3 has blocks along the diagonal: for each $a_2 \in A_2$, there is a set of constraints (one constraint for every $a_2' \in A_2$) whose only nonzero coefficients correspond to the variables $p_{(a_1,a_2)}$ (one variable for every $a_1 \in A_1$). The exception is the probability constraint which has nonzero coefficients for all variables (cf. Dantzig-Wolfe decomposition, Dantzig and Wolfe 1960.) The following proposition will help us to understand the relationship to the multiple-LPs approach, and hence the correctness of Linear Program 3.

Proposition 1. *Linear Program 3 always has an optimal solution in which only a single block of variables takes nonzero values. That is, there exists an optimal solution for which there is some $a_2^* \in A_2$ such that for any $a_1 \in A_1, a_2 \in A_2$ where $a_2 \neq a_2^*$, $p_{(a_1,a_2)} = 0$.*

Proof. Suppose for the sake of contradiction that all optimal solutions require nonzero values for at least k blocks, where $k \geqslant 2$. For an optimal solution p with exactly

k nonzero blocks, let $a_2, a_2' \in A_2$, $a_2 \neq a_2'$ be such that $t_{a_2} = \sum_{a_1 \in A_1} p_{(a_1,a_2)} > 0$ and $t_{a_2'} = \sum_{a_1 \in A_1} p_{(a_1,a_2')} > 0$. Let $v_{a_2} = \sum_{a_1 \in A_1} p_{(a_1,a_2)} u_1(a_1, a_2)$ and $v_{a_2'} = \sum_{a_1 \in A_1} p_{(a_1,a_2')} u_1(a_1, a_2')$. Without loss of generality, suppose that $v_{a_2}/t_{a_2} \geqslant v_{a_2'}/t_{a_2'}$. Then consider the following modified solution p':

- for all $a_1 \in A_1$, $p'_{(a_1,a_2)} = \frac{t_{a_2}+t_{a_2'}}{t_{a_2}} p_{(a_1,a_2)}$;
- for all $a_1 \in A_1$, $p'_{(a_1,a_2')} = 0$;
- for all $a_2'' \notin \{a_2, a_2'\}$, $p'_{(a_1,a_2'')} = p_{(a_1,a_2'')}$.

p' has $k-1$ blocks with nonzero values; we will show that p' remains feasible and has at least the same objective value as p, and must therefore be optimal, so that we arrive at the desired contradiction.

To prove that p' is still feasible, we first notice that any of the constraints corresponding to the unchanged blocks (for $a_2'' \notin \{a_2, a_2'\}$) must still hold because none of the variables with nonzero coefficients in these constraints have changed value. The constraints for the block corresponding to a_2' hold trivially because all the variables with nonzero coefficients are set to zero. The constraints for the block corresponding to a_2 still hold because all the variables with nonzero coefficients have been multiplied by the same constant $\frac{t_{a_2}+t_{a_2'}}{t_{a_2}}$. Finally, the probability constraint still holds because the total probability on the variables in the a_2 block is $\sum_{a_1 \in A_1} p'_{(a_1,a_2)} = \sum_{a_1 \in A_1} \frac{t_{a_2}+t_{a_2'}}{t_{a_2}} p_{(a_1,a_2)} = \frac{t_{a_2}+t_{a_2'}}{t_{a_2}} t_{a_2} = t_{a_2} + t_{a_2'}$, that is, we have simply shifted the probability mass from a_2' to a_2. (All the probabilities are also still nonnegative, because $\frac{t_{a_2}+t_{a_2'}}{t_{a_2}}$ is positive.)

To prove that p' is no worse than p, we note that the total objective value derived under p' from variables in the a_2 block of variables is $\sum_{a_1 \in A_1} p'_{(a_1,a_2)} u_1(a_1, a_2) = \sum_{a_1 \in A_1} \frac{t_{a_2}+t_{a_2'}}{t_{a_2}} p_{(a_1,a_2)} u_1(a_1, a_2) = \frac{t_{a_2}+t_{a_2'}}{t_{a_2}} v_{a_2} \geqslant v_{a_2} + v_{a_2'}$, where the inequality follows

from $v_{a_2}/t_{a_2} \geqslant v_{a_2'}/t_{a_2'}$. On the other hand, the total objective value derived under p' from variables in the a_2' block of variables is 0 because all these variables are set to zero. In contrast, under the solution p, the total objective value from these two blocks is $v_{a_2} + v_{a_2'}$. Because p and p' agree on the other blocks, it follows that p' obtains at least as large an objective value as p, and we have a contradiction. □

Proposition 1 suggests that one approach to solving Linear Program 3 is to force all the variables to zero with the exception of a single block and solve the remaining linear program; we try this for every block, and take the optimal solution overall. However, this approach coincides exactly with the original multiple-LPs approach, because:

Observation 1. *In Linear Program 3, if for some a_2, we force all the variables $p_{(a_1,a_2')}$ for which $a_2' \neq a_2$ to zero, then the linear program that remains is identical to Linear Program 1.*

This also proves the correctness of Linear Program 3 (because the multiple-LPs approach is correct).

2.3 Game-theoretic Interpretation: Commitment to Correlated Strategies

Linear Program 3 can be interpreted as follows. Player 1 commits to a *correlated* strategy. This entails that player 1 chooses a distribution $p_{(a_1,a_2)}$ over the outcomes, and commits to acting as follows: she draws (a_1, a_2) according to the distribution, recommends to player 2 that he should play a_2, and plays a_1 herself. The constraints

$(\forall a_2, a_2' \in A_2)$

$\sum_{a_1 \in A_1} p_{(a_1,a_2)} u_2(a_1, a_2') \leqslant \sum_{a_1 \in A_1} p_{(a_1,a_2)} u_2(a_1, a_2)$

in Linear Program 2 then mean that player 2 should always follow the recommendation a_2 rather than take some alternative action a_2'. This is for the following reasons:

if for some a_2, $\sum_{a_1 \in A_1} p_{(a_1,a_2)} = 0$, then there will never be a recommendation to player 2 to play a_2, and indeed the constraint will hold trivially in this case. On the other hand, if $\sum_{a_1 \in A_1} p_{(a_1,a_2)} > 0$, then player 2's subjective probability that player 1 will play a_1 given a recommendation of a_2 is $P(a_1|a_2) = \frac{p_{(a_1,a_2)}}{\sum_{a_1' \in A_1} p_{(a_1',a_2)}}$. Hence player 2 will be incentivized to follow the recommendation of playing a_2 rather than a_2' if and only if $(\forall a_2, a_2' \in A_2) \sum_{a_1 \in A_1} P(a_1|a_2) u_2(a_1, a_2') \leqslant \sum_{a_1 \in A_1} P(a_1|a_2) u_2(a_1, a_2)$ which is identical to the constraint in Linear Program 3 (by multiplying by $\sum_{a_1' \in A_1} p_{(a_1',a_2)}$).

Proposition 1 entails that we can without loss of optimality restrict attention to solutions where the recommendation to player 2 is always the same (so that there is no information in the signal to player 2).

Given this interpretation of Linear Program 3, it is not surprising that it is very similar to the linear feasibility formulation of the correlated equilibrium problem for two players.

Linear Program 3 is identical to Linear Program 2, except that in Linear Program 3 we have dropped the incentive constraints for player 1, and added an objective of maximizing player 1's expected utility. If we add the incentive constraints for player 1 back in, then we obtain a linear program for finding the correlated equilibrium that maximizes player 1's utility. Because adding constraints cannot increase the objective value of a maximization problem, we immediately obtain the following corollary:

Corollary 1 (von Stengel and Zamir 2010)**.** *Player* 1*'s expected utility from optimally committing to a mixed strategy is at least as high as her utility in any correlated equilibrium of the simultaneous-move game.*

2.4 Commitment to Correlated Strategies with More Players

We have already seen that committing to a correlated strategy in a two-player game is in some sense not particularly interesting, because without loss of optimality player 2 will always get the same recommendation from player 1. However, the same is not true for games with $n \geqslant 3$ players, where player 1 commits to a correlated strategy and sends recommendations to players $2, \ldots, n$, who then play simultaneously. We can easily extend Linear Program 3 to this case of n players (just as it is well known that Linear Program 2 can be extended to the case of n players):

Linear program 4.

$$\max \sum_{a_1 \in A_1, \ldots, s_n \in S_n} p_{(a_1,\ldots,s_n)} u_1(a_1, \ldots, s_n)$$

subject to:

$$(\forall i \in \{2, \ldots, n\})\ (\forall s_i, s_i' \in S_i)$$

$$\sum_{s_{-i} \in S_{-i}} p_{(s_i, s_{-i})} u_i(s_i', s_{-i}) \leqslant \sum_{s_{-i} \in S_{-i}} p_{(s_i, s_{-i})} u_i(s_i, s_{-i})$$

$$\sum_{a_1 \in A_1, \ldots, s_n \in S_n} p_{(a_1,\ldots,s_n)} = 1$$

$$(\forall a_1 \in A_1, \ldots, s_n \in S_n)\ p_{(a_1,\ldots,s_n)} \geqslant 0$$

(Here, we followed the standard game theory notation of using $-i$ to refer to the players other than i.) Again, Linear Program 4 is simply the standard linear feasibility program for correlated equilibrium, with the constraints for player 1 omitted and with an objective of maximizing player 1's expected utility. This immediately implies the following proposition:

Proposition 2. *The optimal correlated strategy to commit to in an n-player normal-form game can be found in time polynomial in the size of the input.*

The following example illustrates that if there are three or more players, then com-

mitment to a correlated strategy can be strictly better for player 1 than commitment to a mixed strategy (as well as any correlated equilibrium of the simultaneous-move version of the game).

Example 1. *Consider a three-player game between a wildlife refuge Manager (player* 1*, aka. M), a Lion (player* 2*, aka. L), and a wildlife Photographer (player* 3*, aka. P). There are four locations in the game: A and B (two locations in the refuge that are out in the open), C (a safe hiding place for the lion), and D (the wildlife photographer's home). Each player must choose a location: M can choose between A and B, L between A, B, and C, and P between A, B, and D.*

M wants L to come out into the open, and would prefer even more to be in the same place as L in order to study him. Specifically, M gets utility 2 *if she is in the same location as L,* 1 *if L is at A or B but not at the same location as M, and* 0 *otherwise. L just wants to avoid contact with humans. Specifically, L gets utility* 1 *unless he is in the same location as another player, in which case he gets* 0*. P wants to get a close-up shot of L, but would rather stay home than go out and be unsuccessful. Specifically, P gets utility* 2 *for being in the same location as L, and otherwise* 1 *for being at D, and* 0 *otherwise.*

A correlated strategy specifies a probability for every outcome, that is, every feasible triplet of locations for the players. We will show that the unique optimal correlated strategy for M to commit to is: $p_{(A,B,D)} = 1/2$*,* $p_{(B,A,D)} = 1/2$*. That is, she flips a coin to determine whether to go to A or B, signals to L to go to the other location of the two, and always signals to P to stay home at D. This is not a correlated equilibrium of the simultaneous-move game, because M would be better off going to the same location as L. This does not pose a problem because M is* committing *to the strategy. The other two players are best-responding by following the recommen-*

dations: L is getting his maximum possible utility of 1*; P (whose signal is always D and thus carries no information) is getting* 1 *for staying home, and switching to either A or B would still leave her with an expected utility of* 1*.*

To see that M cannot do better, note that L can guarantee himself a utility of 1 *by always choosing C, so there is no feasible solution where L has positive probability of being in the same location as another player. Hence, in any feasible solution, any outcome where M gets utility* 2 *has zero probability. All that remains to show is that there is no other feasible solution in which M always gets utility* 1*. In any such solution, L must always choose A or B. Also, in any feasible solution, P cannot play A or B with positive probability, because she can never be in the same location as L; hence, if she played A or B with positive probability, she would end up with an expected utility strictly below* 1*, whereas she can guarantee herself* 1 *just by choosing D. Because M also cannot be in the same location as L with positive probability, it follows that only* $p_{(A,B,D)}$ *and* $p_{(B,A,D)}$ *can be set to positive values. If one of them is set to a value greater than* $1/2$*, then P would be better off choosing the location where L is more than half the time. It follows that* $p_{(A,B,D)} = 1/2$*,* $p_{(B,A,D)} = 1/2$ *is the unique optimal solution.*

2.5 Experiments

While Linear Programs 3 and 4 are valuable from the viewpoint of improving our conceptual understanding of commitment, it is also worthwhile to investigate them as an algorithmic contribution. Linear Program 4 allows us to do something we could not do before, namely, to compute an optimal correlated strategy to commit to in games with more than two players. This cannot be said about the special case of Linear Program 3, because we already had the multiple-LPs approach. But how does Linear Program 3 compare to the multiple-LPs approach? At least, it provides a slight implementation advantage, in the sense of not having to write code to iterate

through multiple LPs. More interestingly, what is the effect on runtime of using it rather than the multiple-LPs approach? Of course, this depends on the LP solver. Does the solver benefit from having the problem decomposed into multiple LPs? Or does it benefit from seeing the whole problem at once?

To answer these questions, we evaluated our approach on GAMUT (Nudelman et al., 2004), which generates games according to a variety of distributions. Focusing on two-player games, we used CPLEX 10.010 both for the multiple-LPs approach (LP1) and for Linear Program 3 (LP3). For each GAMUT game class, we generated 50 two-player games with 50 strategies per player and compared the time it takes to find the optimal strategies to commit to in these games using LP1 and LP3. We show the boxplots of the run times in Figure 2.1. Perhaps surprisingly, it turns out that LP3 generally solves much faster. One possible explanation for this is as follows. In the multiple-LPs approach, each block is solved to optimality separately. In contrast, when presented with LP3, the solver sees the entire problem instance all at once, which in principle could allow it to quickly prune some blocks as being clearly suboptimal. The only distribution for which LP3 is slower is RandomZeroSum. Unfortunately, preliminary experiments on random games indicate that LP3 does not scale gracefully to larger games, and that perhaps LP1 scales a little better. We conjecture that this is due to heavier memory requirements for LP3.

Table 2.1 shows how often the correlated strategy to commit to computed by Linear Program 4 is a product distribution. We say that a distribution $p(a_1, \ldots, s_n)$ is a product distribution iff it satisfies the following condition.

$$\forall i \in \{1, \ldots, n\}$$
$$\forall a_i \in \{a_i'' \in A_i : p(a_i'') > 0\}$$
$$\forall a_i' \in \{a_i'' \in A_i : p(a_i'') > 0\}$$
$$\forall s_{-i} \in S_{-i} : p(s_{-i}|a_i) = p(s_{-i}|a_i')$$

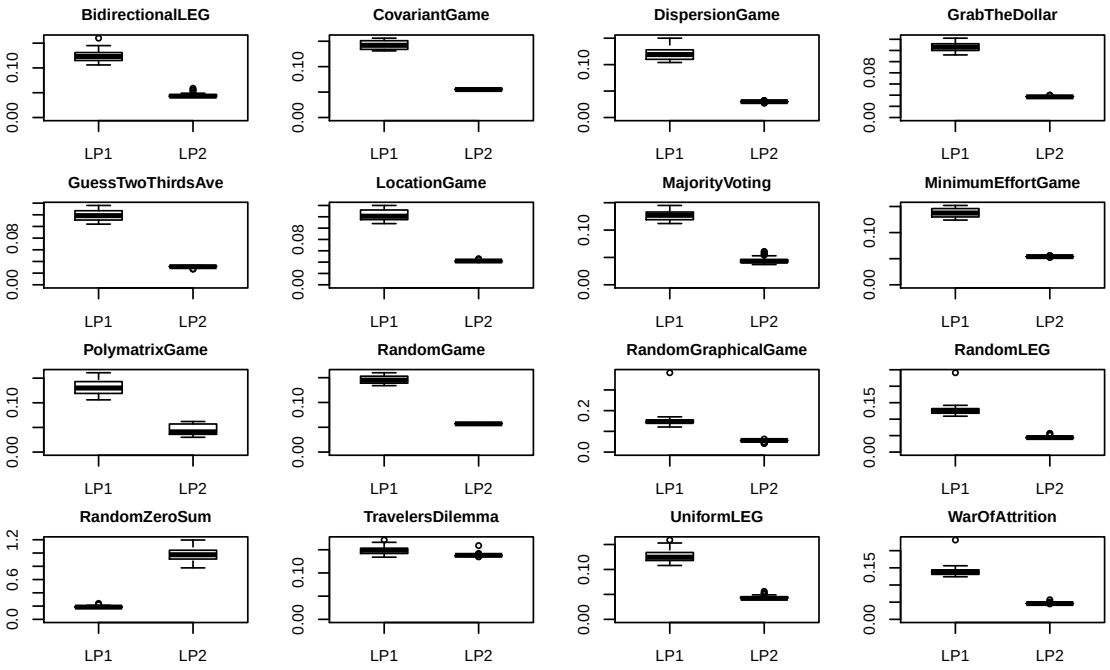

FIGURE 2.1: Run time comparison of LP1 and LP3 on GAMUT games (seconds).

Low percentages here indicate that correlation plays a significant role. To compute this data, we generated 50 payoff matrices with 10 strategies per player for each combination of a GAMUT class and a number of players. In other words, if the leader commits to a product distribution over the strategy profiles, then the recommendation each of the followers gets from the correlated strategy does not give out any information about the recommendations that the other players receive. In games with two players, the correlated strategy computed by LP3 is always a product distribution, as expected by Proposition 1. For games with more than two players, in some distributions correlation does not play a big role, and in others it does.

We say that a correlated strategy is a degenerate distribution if its support size is 1. A degenerate distribution is a special case of a product distribution. As we can see from Table 2.1, a large fraction of computed product distributions are actually degenerate.

Table 2.1: For each game class and number of players, the two numbers shown are the fractions of product distributions (P) and the fractions of degenerate distributions (D) among the correlated strategies computed by LP4.

Game class \ ♯ players	**2**		**3**		**4**	
	P	D	P	D	P	D
BidirectionalLEG	1	.96	.9	.86	.84	.84
CovariantGame	1	.48	.64	.6	.68	.68
DispersionGame	1	1	1	1	1	1
GuessTwoThirdsAve	1	1	0	0	0	0
MajorityVoting	1	.88	1	1	1	1
MinimumEffortGame	1	1	1	1	1	1
RandomGame	1	.42	.16	.08	.02	.02
RandomGraphicalGame	1	.4	.22	.1	.02	.02
RandomLEG	1	1	.92	.92	.02	.02
TravelersDilemma	1	0	1	1	.02	.02
UniformLEG	1	.96	.88	.86	.02	.02

In the next chapter, we will consider another extension of the leader-follower model. We will consider the case in which the players are uncertain about whether the follower can observe the leader's strategy.

3

Solving Stackelberg Games with Uncertain Observability

Playing a Stackelberg strategy seems to make little sense without some argument as to why the player should indeed be able to commit before her opponent moves. In the real-world security applications mentioned in the introduction, where Stackelberg strategies are indeed used, the argument is that the attacker (follower) can observe the defender (leader)'s actions over time, and thereby reconstruct the distribution, before attacking. This argument is not entirely uncontroversial: in many contexts, it is not clear that the follower can indeed observe the leader's mixed strategy. A recent study shows that a large class of security games has the property that any Stackelberg strategy is also a Nash equilibrium strategy, and moreover that there is no equilibrium selection problem (Korzhyk et al., 2011). Nevertheless, this is known to not be true for other types of security games.

How should the leader agent play when she is not sure about the follower's ability to observe her mixed strategy, as is often the case in practice? One model that has been proposed in the paper by Korzhyk et al. (2011) for this is to consider

an extensive-form game where Nature makes a random move determining whether the leader's mixed strategy is observable or not, and then to find an equilibrium of this larger game. We will discuss this model in detail in Section 3.1. In this chapter, we study properties of this model, present the first algorithm for solving these infinite-size extensive-form games, and evaluate it on random games. Our algorithm calls subroutines for solving Nash and Stackelberg problems; it works for any game representation (as long as the Nash and Stackelberg subroutines do).

3.1 Review: Extensive-Form Game to Model Uncertainty about Observability

There are two players in the original game (represented in normal form): the leader and the follower. The leader's set of pure actions is A_l. The follower's set of pure actions is A_f. If the outcome of the game is (a_l, a_f), where $a_l \in A_l$ is the leader's action and $a_f \in A_f$ is the follower's action, then the leader's utility is $u_l(a_l, a_f)$, and the follower's utility is $u_f(a_l, a_f)$.

We now present the extensive-form game model introduced in the paper by Korzhyk et al. (2011) which is arguably the most straightforward way to introduce uncertainty about the follower's ability to observe the leader's distribution over A_l.

Let us first explain what is the *extensive form*. We have seen the normal-form representation of games in the introduction. The extensive form is a more expressive representation which can be used to describe general games. Each player is represented by one or more nodes at which the player can choose her actions. The actions correspond to the edges connected to the player's nodes. The sequence in which the players choose their actions is specified by the path which starts at the root and follows along the edges corresponding to the actions chosen by the players until a leaf is reached. If several nodes of a player are in the same *information set*, then the player cannot distinguish between those nodes. Some of the nodes may belong to

Nature, which chooses its actions at random.

The extensive-form game with uncertainty about observability shown in Figure 3.1 proceeds as follows. First, Nature decides whether the follower will observe the leader's distribution or not. The probability that the follower observes the leader's distribution is p_{obs}; correspondingly, the probability that the follower does not observe it is $1 - p_{\text{obs}}$. Then, the leader, without knowing Nature's choice, chooses a distribution over A_l. (Note that there are infinitely many distributions to choose from—in particular, choosing a distribution is not the same as randomizing over which action to choose here.) Next, the follower chooses a response $a_f \in A_f$, possibly after observing the distribution over A_l chosen by the leader if Nature has decided that the follower is able to observe.

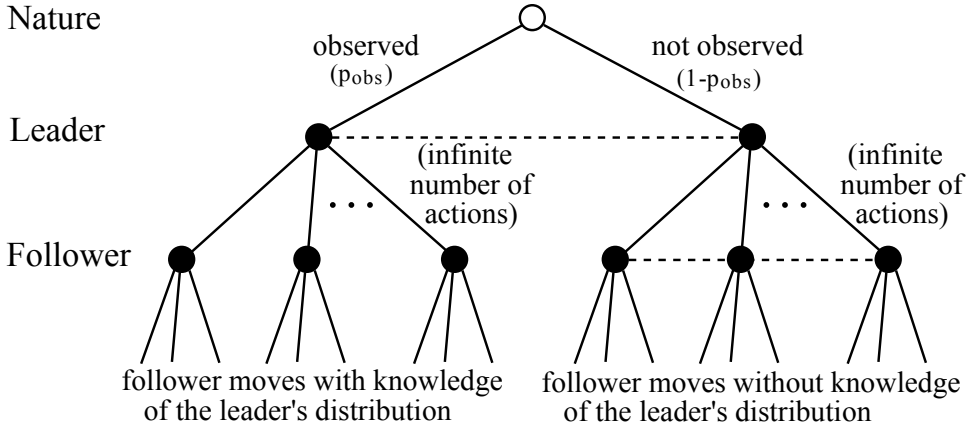

FIGURE 3.1: The extensive form of the game.

Nodes that are in the same information set are connected with dashed lines. The two leader nodes are in the same information set because the leader does not observe Nature's decision. The follower's nodes in the right subtree are in the same information set, because the right subtree corresponds to the case where the follower does not observe the distribution.

It is important to emphasize that a *pure* strategy for the leader in this extensive-form game is a *distribution* over A_l; a mixed strategy for the leader is a distribution

over such distributions. (In fact, we will show shortly that a distribution over distributions over A_l cannot be simplified to a distribution over A_l in this context.) A pure strategy for the follower specifies one action in A_f for every follower node on the left-hand side of the tree, plus one additional action for the follower's information set on the right-hand side of the tree. In fact, it is possible to simplify the left-hand side of the tree: we can take the follower's best action at each of his nodes on the left-hand side, and simply propagate the corresponding value up to that node as in backward induction.[1] (If there is a tie for the follower, he will break it in favor of the leader, to stay consistent with the Stackelberg model.) Thus we can eliminate the bottom level of the left-hand side of the tree, so that effectively a follower pure strategy in the extensive form consists of only a single action in A_f, corresponding to his action in the information set on the right-hand side.

Since our goal is to solve an extensive-form game, a natural question is whether off-the-shelf extensive-form game solvers are sufficient for this. As we have pointed out, the leader's strategy space is infinite, preventing the direct application of standard methods. One way to address this is to discretize the leader's strategy space and obtain an approximate solution. Because this strategy space is an $(|A_l| - 1)$-simplex, discretizing it sufficiently finely is likely to lead to scalability issues. Our algorithm, in contrast, generates pure strategies for the leader in an informed way that results in an exact solution. Moreover, as we will see, experimentally our algorithm requires the generation of only very few strategies, so that there can be little doubt that this is preferable to the uninformed discretization approach.

[1] Note that we are just doing this at a conceptual level; we never actually write down this (infinite-sized) tree.

3.2 Equilibria May Require Randomizing over Distributions

Because pure strategies for the leader in the extensive-form game are distributions over A_l, it follows that mixed strategies for the leader are distributions over distributions. However, one may be skeptical as to whether it is ever really necessary to randomize over distributions, rather than just simplifying the strategy back down to a single distribution. In this subsection, we show that for some games, randomizing over distributions is in fact necessary, in the sense that there is no equilibrium of the extensive-form game in which the leader plays a pure strategy.

	EL	L	R	ER
U	9,10	0,9	1,8	10,0
D	10,0	1,8	0,9	9,10

FIGURE 3.2: An example normal-form game.

Consider the example game in Figure 3.2. This game has no pure-strategy Nash equilibrium. The unique mixed-strategy Nash equilibrium profile of this game is $\langle (0.5, 0.5), (0, 0.5, 0.5, 0) \rangle$.[2] The row player's utility from playing this equilibrium is 0.5. In contrast, in the Stackelberg model, the row player can commit to playing U, so that the column player best-responds with EL, which results in a utility of 9 for the row player. The row player can achieve an even higher utility by committing to a mixed strategy. If the row player commits to playing U with probability $8/9 + \epsilon$ and D with probability $1/9 - \epsilon$, the column player's best-response is still EL, and the row player's utility is approximately $9 + 1/9$. The Stackelberg solution is the limit as $\epsilon \to 0$. (Note that there are symmetric solutions on the other side of the game where

[2] The equilibrium is unique because of the following. If the row player plays U with probability ¿ 0.5, then only EL and L can be best responses for the column player, but then U cannot be a best response for the row player. By symmetry, the row player also cannot play D with probability ¿ 0.5. Hence any equilibrium has the row player playing $(0.5, 0.5)$. Only L and R are best responses to this for the column player, and the only way to put probability on these to keep the row player indifferent between U and D is $(0, 0.5, 0.5, 0)$.

the row player puts most of the probability on D and the column player responds with ER.)

Now consider the extensive-form variant of this game where the leader's (row player's) distribution is observed with probability $p_{\text{obs}} = .99$. Because the leader's distribution is almost always observed, it is suboptimal for the leader to put positive probability on any distribution that has probability strictly between 1/9 and 8/9 on U. This is because, when observed (which happens almost always), such distributions would incentivize the follower to play L or R, whereas any more extreme distributions will incentivize the follower to play EL or ER, leading to much higher utilities for the leader. (We recall that, upon observing the distribution, the follower is assumed to break ties in the leader's favor for technical reasons, though this is not essential for the example.)

It is also suboptimal to put positive probability on any distribution that puts strictly more than 8/9 probability on U. This is because, as long as the probability on U is at least 8/9, any unit of probability mass placed on D results in a utility of 10 rather than 9 in the .99 of cases where the follower observes; this outweighs any benefit that placing this unit of probability elsewhere might have in the .01 of cases where the follower does not observe. Similarly, putting positive probability on any distribution that puts strictly less than 1/9 probability on U is suboptimal. Hence, all of the leader's mass is either on the distribution $(8/9, 1/9)$ or on the distribution $(1/9, 8/9)$.

If the leader places all her mass on the distribution $(8/9, 1/9)$, the follower is incentivized to play EL all the time. However, if this is so, the leader has an incentive to deviate to $(1/9, 8/9)$. This is because this distribution will give her just as high a utility as $(8/9, 1/9)$ if it is observed (the follower will respond with ER); however, if it is not observed, the follower will not know that the leader has deviated and still play EL, and $(1/9, 8/9)$ gives a higher utility against EL than $(8/9, 1/9)$. Hence there is

no equilibrium where the leader places all her mass on $(8/9, 1/9)$ (and, by symmetry, there is none where the leader places all her mass on $(1/9, 8/9)$). In fact, by similar reasoning as that used to establish the uniqueness of the Nash equilibrium of the original game, we can conclude that in equilibrium the leader must randomize uniformly between $(8/9, 1/9)$ and $(1/9, 8/9)$; the follower must then respond accordingly with EL or ER when he observes the distribution, and when he does not observe the distribution he must randomize uniformly between L and R (to keep the leader indifferent between her two distributions). Hence, this is the unique equilibrium.

3.3 The Algorithm

We now present our algorithm for solving for an equilibrium of the extensive-form game (Figure 3.1). The intuition behind the algorithm is as follows. As we have already pointed out, after applying backward induction to the left-hand side of the extensive-form game, the follower's pure strategy space in the extensive-form game is simply A_f (corresponding to the action he takes on the right-hand side), which is manageable. What is not manageable is the space of all the leader pure strategies in the extensive form: there is one for every distribution over A_l, so there are infinitely many. This prevents us from simply writing down the normal-form game corresponding to the extensive-form game and solving that. (Note that this is *not* the same as the original normal-form game that has no uncertainty about observability.)

To address this, we start with a limited set of leader distributions (for example, the set of all $|A_l|$ degenerate distributions), and solve for a Nash equilibrium of this restricted game. This will give us a mixed strategy for the follower; the next step is to find the best leader pure strategy (distribution over A_l) in response to this follower mixed strategy. As we will see, technically, this corresponds to solving for a Stackelberg solution of an appropriately modified normal-form game. We then add the resulting distribution to the set of leader distributions, solve for a new

equilibrium, etc., until convergence.

This type of strategy generation approach has been applied to solve various games where the strategy space is too large to write down (McMahan et al., 2003; Halvorson et al., 2009; Jain et al., 2010). (It has a close relation to the notion of constraint / column generation in linear programming.) Usually, this is because the strategy space is combinatorial—but it is finite, and hence the algorithm is guaranteed to converge eventually. In our case, however, there is a continuum of leader strategies, so we have to prove convergence, which we will do later.

Our algorithm for finding an equilibrium of the extensive-form game is shown in Figure 3.3. In this algorithm, $\mathcal{G}(D, A_f)$ is a normal-form game, more specifically it is the normal-form game corresponding to the extensive-form game, except that the leader can only choose from the distributions in D.

$D \leftarrow$ *any finite non-empty set of distributions over* A_l
Loop:
  $\mathcal{G} \leftarrow \mathcal{G}(D, A_f)$
  $\langle s_l^{\mathcal{G}}, s_f \rangle \leftarrow$ FIND-NE($\mathcal{G}$)
  $s_l' \leftarrow$ LEADER-BR(s_f)
  If $u_l^{\mathcal{G}}(s_l', s_f) \leqslant u_l^{\mathcal{G}}(s_l^{\mathcal{G}}, s_f)$ **Then**
    Return $\langle s_l^{\mathcal{G}}, s_f \rangle$
  Else
    $D \leftarrow D \cup \{s_l'\}$

FIGURE 3.3: The algorithm.

At any point, D is the set of distributions for the leader that we have generated so far. We find a mixed-strategy Nash equilibrium $\langle s_l^{\mathcal{G}}, s_f \rangle$ of a normal-form game $\mathcal{G}$ in which the leader's set of pure strategies is D, the follower's set of pure strategies

is A_f, and the players' utilities for the outcome (d, a_f) are defined as follows.

$$u_l^{\mathcal{G}}(d, a_f) = p_{\text{obs}}\mathbb{E}_{a_l \sim d}[u_l(a_l, \text{FOLLOWER-BR}_{\text{obs}}(d))] + (1 - p_{\text{obs}})\mathbb{E}_{a_l \sim d}[u_l(a_l, a_f)] \tag{3.1}$$

$$u_f^{\mathcal{G}}(d, a_f) = p_{\text{obs}}\mathbb{E}_{a_l \sim d}[u_f(a_l, \text{FOLLOWER-BR}_{\text{obs}}(d))] + (1 - p_{\text{obs}})\mathbb{E}_{a_l \sim d}[u_f(a_l, a_f)] \tag{3.2}$$

Here $d \in D$ is a distribution over A_l; a_l is the leader's action drawn according to d; and $a_f \in A_f$ is the follower's action. u_l and u_f correspond to the utilities in the *original* normal-form game (that did not model uncertain observability). In each of these formulas, the first summand corresponds to the case where the follower observes the leader's chosen distribution over A_l, so that the follower best-responds to that distribution; the second summand corresponds to the case where the follower does not observe the leader's distribution over A_f, so that the follower will follow his strategy a_f for the right-hand side of the extensive-form game. The follower's best-response is computed as follows.

$$\text{FOLLOWER-BR}_{\text{obs}}(d) \in \arg\max_{a_f \in A_f^*} \mathbb{E}_{a_l \sim d}[u_l(a_l, a_f)]$$

$$A_f^* = \arg\max_{a_f \in A_f} \mathbb{E}_{a_l \sim d}[u_f(a_l, a_f)]$$

That is, the follower maximizes his expected utility, breaking the ties in favor of the leader.[3]

We then check whether $s_l^{\mathcal{G}}$ is actually a best-response to s_f if the leader considers all possible distributions over A_l (we only know for sure that it is a best response among the restricted set D). To do that, we compute a best-response distribution s_l' over A_l that maximizes the leader's expected utility $u_d'(s_l', s_f)$. If it turns out

[3] This is a common assumption in Stackelberg games; without it, it may happen that no solution exists. Specifically, if the original normal-form game is generic, then the follower breaks ties in the leader's favor in every subgame-perfect equilibrium of the regular Stackelberg extensive-form game (von Stengel and Zamir, 2010).

that $u'_d(s'_l, s_f)$ is equal to the leader's utility in the computed Nash equilibrium of the game, then it follows that $s_l^{\mathcal{G}}$ is a best response to s_f, and because s_f is also a best response to $s_l^{\mathcal{G}}$, we can return $\langle s_l^{\mathcal{G}}, s_f \rangle$ as an equilibrium of the extensive-form game with uncertain observability. Otherwise, we add distribution s'_l to D, and the algorithm continues on to the next iteration, in which we construct a new game $\mathcal{G}$, compute its Nash equilibrium, and so on.

In Subsection 3.3.1, we show how to compute the leader's best response LEADER-BR(s_f) efficiently using a set of linear programs (corresponding to a Stackelberg solve). In Subsection 3.3.2, we show how the algorithm solves the example game in Figure 3.2 with $p_{\text{obs}} = .99$. In Subsection 3.3.3, we show that the algorithm converges in a finite number of iterations.

3.3.1 Computing the Leader's Best Response

In this section, we describe an efficient way to compute a distribution s'_l over the leader's actions A_l such that the leader's utility of playing s'_l is maximized assuming that the follower plays a given strategy s_f. That is, s'_l is the leader's best response to the follower's mixed strategy s_f, denoted by LEADER-BR(s_f) in the algorithm shown in Figure 3.3.

Our goal is to formulate LEADER-BR as a linear program. However, the leader's utility is not linear in s'_l in the case where the follower observes the leader's mixed strategy, because the leader's utility depends on the follower's best response to this observation, which can be different for different values of s'_l. Hence, we use a trick that is also used in computing Stackelberg strategies (with certain observability) (Conitzer and Sandholm, 2006; von Stengel and Zamir, 2010): we write an LP that maximizes the leader's expected utility under the constraint that the follower's best response in the observed case is a fixed action a_f^*. To find the leader's best response to s_f overall, we solve such an LP for each $a_f^* \in A_f$; we obtain a best response for the leader by

choosing the optimal solution vector s'_l for an LP with the highest objective value (leader utility). Note that some of these LPs may be infeasible.

Specifically, given a^*_f, s_f, we solve the following LP, whose variables are the p'_{a_l}.

$$\text{Maximize } p_{\text{obs}} \sum_{a_l \in A_l} p'_{a_l} u_l(a_l, a^*_f)$$

$$+ (1 - p_{\text{obs}}) \sum_{a_l \in A_l} \sum_{a_f \in A_f} p'_{a_l} q_{a_f} u_l(a_l, a_f)$$

Subject to

$$\forall a_f \in A_f : \sum_{a_l \in A_l} p'_{a_l} u_f(a_l, a^*_f) \geqslant \sum_{a_l \in A_l} p'_{a_l} u_f(a_l, a_f)$$

$$\sum_{a_l \in A_l} p'_{a_l} = 1$$

$$\forall a_l \in A_l : p'_{a_l} \geqslant 0$$

This formulation is almost identical to the standard one for solving for a Stackelberg strategy (Conitzer and Sandholm, 2006; von Stengel and Zamir, 2010), except the objective is different to account for the fact that the follower may not observe the distribution. In fact, if we modify the leader's utility function to $u_l^{s_f}(a_l, a^*_f) = p_{\text{obs}} u_l(a_l, a^*_f) + (1 - p_{\text{obs}}) \sum_{a_f \in A_f} s_f(a_f) u_l(a_l, a_f)$, then the objective simplifies to $\sum_{a_l \in A_l} p'_{a_l} u_l^{s_f}(a_l, a^*_f)$, and we obtain the standard Stackelberg formulation. Hence, we are just doing a Stackelberg solve on a modified game.

3.3.2 An Example Run of the Algorithm

In this section, we demonstrate how the algorithm computes an equilibrium of the uncertain-observability extensive-form game for the payoff matrix shown in Figure 3.2, with probability of observability $p_{\text{obs}} = 0.99$. (We already solved for the equilibrium of this game analytically in Section 3.2—the purpose here is to show how the algorithm finds this equilibrium.) In this game, there are two actions in A_l,

so each leader distribution is represented by a vector of two numbers summing to 1.

Initialization. We initialize the set of leader distributions with the two degenerate distributions over A_l: the distribution $(1, 0)$ corresponds to the leader always playing U, and the distribution $(0, 1)$ corresponds to the leader always playing D. The normal-form game for the current set of distributions $D = \{(1, 0), (0, 1)\}$ and the utilities $u_l^{\mathcal{G}}, u_f^{\mathcal{G}}$ computed according to Equations (3.1), (3.2) is shown in Figure 3.4. (Note that the follower strategy has very little effect on the expected payoffs in this game; this is because the follower strategy only concerns the "unobserved" part of the game, which occurs very rarely in this game. The "observed" part has been preprocessed with backward induction.)

	EL	L	R	ER
(1,0)	9,10	8.91, 9.99	8.92, 9.98	9.01, 9.9
(0,1)	9.01, 9.9	8.92, 9.98	8.91, 9.99	9,10

FIGURE 3.4: The normal-form game after the initialization.

Iteration 1. We first compute a Nash equilibrium of the normal-form game shown in Figure 3.4, namely, $\langle (.5, .5), (0, .5, .5, 0) \rangle$. Next, we compute the leader's best response to the follower's mixed strategy $(0, .5, .5, 0)$. This results in the distribution s_1, in which the leader plays U with probability 8/9 and D with probability 1/9, so that the follower's best response to s_1 is EL.

$$s_1 = (8/9)U + (1/9)D$$

It turns out that the leader's utility from playing s_1 against the follower's mixed strategy $(0, .5, .5, 0)$ is higher than the leader's utility in the current NE profile $\langle (.5, .5), (0, .5, .5, 0) \rangle$. Thus, we add s_1 to D. The resulting normal-form game is shown in Figure 3.5.

Iteration 2. We compute a Nash equilibrium of the game shown in Figure 3.5, namely, the pure-strategy Nash equilibrium $\langle s_1, \mathrm{L} \rangle$. The leader's best response to

	EL	L	R	ER
(1,0)	9,10	8.91, 9.99	8.92, 9.98	9.01, 9.9
(0,1)	9.01, 9.9	8.92, 9.98	8.91, 9.99	9,10
s_1	9.11, 8.89	9.02, 8.89	9.03, 8.88	9.12, 8.81

FIGURE 3.5: The normal-form game after the first iteration.

the follower's strategy L is s_2, where

$$s_2 = (1/9)U + (8/9)D$$

The leader's utility from playing s_2 against L is higher than the leader's utility from playing s_1 against L. Thus, we add s_2 to the set D. The resulting normal-form game is shown in Figure 3.6.

	EL	L	R	ER
(1,0)	9,10	8.91, 9.99	8.92, 9.98	9.01, 9.9
(0,1)	9.01, 9.9	8.92, 9.98	8.91, 9.99	9,10
s_1	9.11, 8.89	9.02, 8.89	9.03, 8.88	9.12, 8.81
s_2	9.12, 8.81	9.03, 8.88	9.02, 8.89	9.11, 8.89

FIGURE 3.6: The normal-form game after the second iteration.

Iteration 3. We compute a mixed-strategy Nash equilibrium of the normal-form game shown in Figure 3.6, namely, $\langle (0, 0, .5, .5), (0, .5, .5, 0) \rangle$. When we compute the leader's best-response to the follower's mixed strategy $(0, .5, .5, 0)$, it turns out that there is no distribution that gives the leader a utility higher than the leader's utility in the computed NE profile. Thus we have found an equilibrium of the uncertain-observability extensive-form game, in which the leader plays s_1 with probability .5 and s_2 with probability .5, while the follower plays L with probability .5 and R with probability .5.

3.3.3 A Bound on the Number of Iterations

In this section, we prove that the algorithm is guaranteed to find an equilibrium of the extensive-form game in a finite number of iterations. For each a_f, the set of leader mixed strategies S^{a_f} to which a_f is a best response is a polytope in $\mathbb{R}^{|A_l|}$. Denote the number of vertices of S^{a_f} by $v(S^{a_f})$. Typical linear program solvers will return a vertex of the feasible region; we will assume that we use such a solver. Then, the number of iterations of our algorithm can be bounded as follows.

Theorem 1. *The algorithm finds an equilibrium of the extensive-form game modeling uncertain observability in no more than* $1 + \sum_{a_f \in A_f} v(S^{a_f})$ *iterations.*

Proof. LEADER-BR returns the optimal solution to one of the linear programs in Subsection 3.3.1. The feasible region of each of these linear programs is one of the regions S^{a_f}. Hence, by the assumption on our LP solver, LEADER-BR always returns a vertex of such a region.

When we generate a vertex corresponding to a distribution that is already in D, we have converged: this vertex cannot be a better response to s_f than s_l, because s_l is a best response to s_f among distributions in D. Because there are at most $\sum_{a_f \in A_f} v(S^{a_f})$ distinct vertices to generate, the bound on the number of iterations follows. □

3.4 A Stronger Bound on the Leader's Support Size

Theorem 1 implies that there always exists an equilibrium in which the leader randomizes over at most $1 + \sum_{a_f \in A_f} v(S^{a_f})$ distributions. This is still a rather loose bound. The following theorem establishes a much tighter bound.

Theorem 2. *In any uncertain-observability extensive-form game, there exists an equilibrium in which the number of distributions on which the leader places positive*

probability is at most $|A_l|$.

Proof. Let s_l denote a distribution over leader actions, where $s_l(a_l)$ denotes the probability s_l places on leader action $a_l \in A_l$. Suppose there is an equilibrium of the whole game with $s_l^{\mathcal{G}}(s_l)$ denoting the leader probability on distribution s_l, and $s_f(a_f)$ denoting the follower probability on follower action a_f (conditional on the follower not being able to observe). Let $\pi(a_l) = \sum_{s_l} s_l^{\mathcal{G}}(s_l)s_l(a_l)$ be the marginal probability that the leader plays a_l. Finally, let $u_l^s(s_l)$ denote the utility that the leader would get for committing to s_l in a pure Stackelberg version of the game (corresponding to the "observed" side of the game tree). Then, consider the following linear program whose variables are p'_{s_l} (one for every distribution s_l in the support of $s_l^{\mathcal{G}}$). (This LP is just for the purpose of analysis.)

$$\text{Maximize } \sum_{s_l} p'_{s_l} u_l^s(s_l)$$

$$\text{Subject to}$$

$$(\forall a_l) \sum_{s_l} p'_{s_l} s_l(a_l) = \pi(a_l)$$

$$(\forall s_l)\ p'_{s_l} \geqslant 0$$

That is, this linear program tries to modify the leader's equilibrium strategy to maximize the leader's overall Stackelberg utility (the utility on the "observed" side of the game tree) under the constraint that the marginal probabilities do not change (so that nothing changes on the "unobserved" side of the tree).

The original equilibrium strategy $s_l^{\mathcal{G}}$ must be an optimal solution to this LP, because, if we suppose to the contrary that there is a better solution, then the leader would want to switch to that better solution (it would not change her utility on the "unobserved" side and it would improve it on the "observed" side), contradicting the equilibrium assumption. In fact, any optimal solution to this linear program must

be an equilibrium when combined with the s_f, because it will do just as well as $s_l^{\mathcal{G}}$ for the leader, and the follower will still be best-responding (on the "unobserved" side) because the marginal probabilities on the a_l remain the same. A linear program with $|A_l|$ constraints (not counting the nonnegativity constraints for each variable) must have an optimal solution with at most $|A_l|$ of its variables set to nonzero values (which follows, for example, from the simplex algorithm). It follows that there exists an equilibrium where the leader places positive probability on at most $|A_l|$ distributions. □

3.5 Experiments

The goal of our experiments is to study a number of properties of the proposed algorithm and the solutions it generates. Since the bound on the number of iterations given in Theorem 1 is quite loose, we want to measure the number of iterations and the overall run time of the algorithm for different payoff matrices and values of p_{obs}. Another goal of the experiments is to measure the leader's support size, that is, the number of distributions played with positive probability in the leader's equilibrium strategy, which we showed to be bounded by the number of the leader's actions $|A_l|$ (Theorem 2). We also want to study the dependence of the leader's equilibrium utility on the probability of observability p_{obs}. Finally, we want to find out how often the leader's equilibrium strategy in the extensive-form game is actually different from Nash and Stackelberg strategies in the original normal-form game.

In our experimental results we consider 15×15 payoff matrices and vary p_{obs}. We used two different Nash equilibrium solvers, a MIP solver with different objectives (Sandholm et al., 2005), and the Gambit (McKelvey et al., 2004) implementation of the Lemke-Howson algorithm (Lemke and Howson, 1964). For the MIP solver, we used three different objective functions: no objective, minimizing the size of the leader support, and maximizing the leader utility.

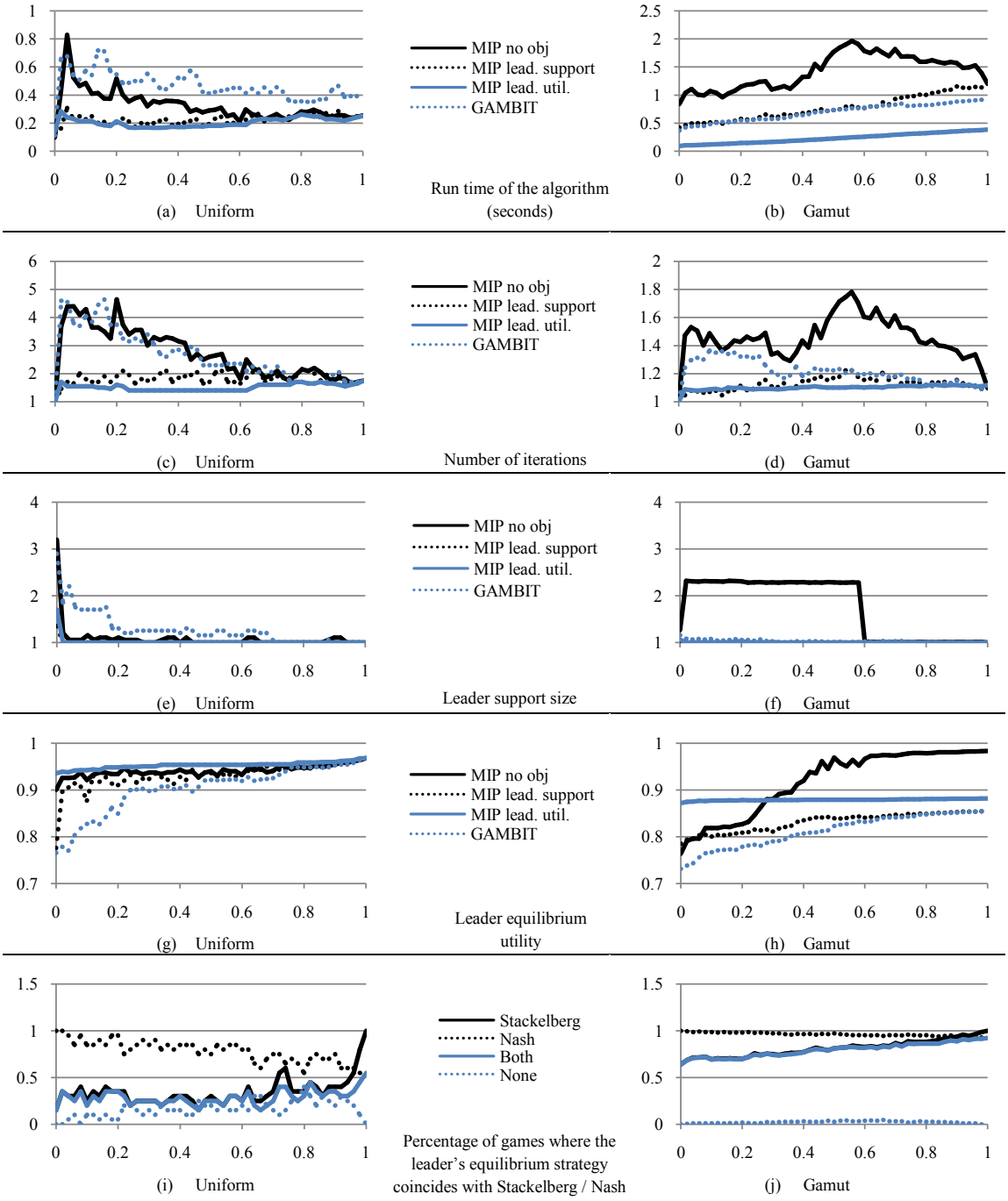

FIGURE 3.7: Experimental results

We considered two distributions over games. The first distribution (*uniform*) generated payoff matrices with individual payoffs drawn uniformly at random from $[0, 1]$. The second (*gamut*) generated payoff matrices from the various game types offered in GAMUT (Nudelman et al., 2004), with uniform weight given to each type.

Figures 3.7(a) and 3.7(b) show the run time of the different algorithms as a function of p_{obs}. One general trend is that the MIP solver that minimizes the leader support is the fastest solver. One interesting difference is that run time generally increases with p_{obs} for the GAMUT distribution, but is fairly flat or decreasing for uniform. The short run time is due to the low number of iterations, which we discuss next.

Figures 3.7(c) and 3.7(d) show the number of iterations taken by the algorithm. Each iteration corresponds to a complete pass through the loop in Figure 3.3, which includes a Nash equilibrium computation in the extensive form game followed by a LEADER-BR solve. The number of iterations generally tracks the run time fairly closely. Two exceptions are GAMBIT and MIP with leader support minimization for the GAMUT distribution. As we can see, the number of iterations is surprisingly low compared to our theoretical bound of Theorem 1. We leave the question of whether a tighter theoretical bound on the number of iterations can be obtained for future research.

The support size (number of distributions over which the leader randomizes in the equilibrium) is shown in Figures 3.7(e) and 3.7(f). The small support size is explained in part by the low number of iterations. Since we initialize the algorithm with $|A_l|$ pure strategies for the leader, the leader's support size cannot be larger than $|A_l|$ plus the number of iterations. However, it is significantly lower than that bound.

Figures 3.7(g) and 3.7(h) show the leader's expected utility in the equilibrium. As expected, higher values of p_{obs} lead to higher utility for the leader—this is the benefit

of commitment. Using the MIP that maximizes leader utility (within a single Nash solve) tends to lead to high leader utilities in the final equilibrium, but intriguingly the MIP with no objective surpasses it for the GAMUT games.

Finally, Figures 3.7(i) and 3.7(j) show how often the leader's equilibrium strategy coincided with Stackelberg (full observability) or Nash (no observability) strategies of the game. The Nash subroutine that is used by the algorithm here is the MIP formulation that minimizes the support size. Naturally, the higher the value of p_{obs} is, the more often the equilibrium strategy coincides with Stackelberg and the less often it coincides with Nash. In general, it coincides with Nash very often and with Stackelberg quite often. We can also see that the equilibrium strategy concides with both Nash and Stackelberg at the same time in a high percentage of GAMUT games. This indicates that in certain game families, simply playing a Nash/Stackelberg strategy of the original normal-form game is also an equilibrium strategy in the extensive-form game with uncertain observability across intervals of p_{obs}. However, this is not the case in games with uniformly random payoffs, which suggests the need for an algorithm like the one we present in this paper.

The main lessons that we take away from this set of experiments are as follows. First, our proposed algorithm is quite fast in practice, especially compared to the loose theoretical bound on the number of iterations that we established in Theorem 1. Second, there are games in which the defender's equilibrium strategy is sensitive to the value of p_{obs}, which suggests that it is important to model the uncertainty about the observability. Third, there are families of games in which the equilibrium does not change across wide intervals of p_{obs}—in such cases, playing Nash or Stackelberg strategies of the original normal-form game may be "good enough".

3.6 Discussion

We believe that our algorithm constitutes a useful addition to the toolbox of techniques for computing game-theoretic solutions, especially in ambiguous real-world domains. Strengths of the algorithm include that it can be applied to any game (as opposed to, for instance, just security games), and it can also use as subroutines Nash and Stackelberg solvers that are tailored to particular game families. The algorithm is efficient in practice, and is guaranteed to produce a solution with support no larger than the number of actions in the original game despite solving an extensive form game with a potentially infinite branching factor.

A potential drawback to the overall framework, not the algorithm, is that it requires us to determine the number p_{obs}. This may not be an issue insofar as the solution stays the same across a range of values of p_{obs}, yet many open problems remain. As p_{obs} shrinks, we are more likely to encounter equilibrium selection problems—how do we address these? What happens if we have some degree of control over p_{obs}? Are there other ways of addressing the problem of uncertainty about observability that do not involve making the uncertainty explicit in the extensive form?

In the following chapters, we will consider the simpler models of simultaneous-move games and Stackelberg games. We will focus on the computational complexity of computing Nash equilibria and Stackelberg strategies in security games.

4

Complexity of Computing Optimal Stackelberg Strategies in Security Resource Allocation Games

In this and the following chapters, we consider a class of two-player general-sum games proposed by Kiekintveld et al. (2009). *Security games* have found a number of applications related to homeland security. As was pointed out by Kiekintveld et al. (2009), the applicability of the techniques for computing a Nash equilibrium or a Stackelberg strategy based on the normal form of the game to security domains is limited by the fact that the defender generally has exponentially many pure strategies, so that it is not feasible to write out the entire normal form of the game. Specifically, if there are m indistinguishable defensive resources, and n targets to which they can be assigned ($n > m$), then there are $\binom{n}{m}$ pure strategies (allocations) for the defender. Kiekintveld et al. point out that while the LAX application was small enough to enumerate all strategies, this is not the case for new applications, including the problem of assigning Federal Air Marshals to flights (Tsai et al., 2009). They provide a nice framework for representing this type of problem; we follow this framework in this chapter (and review it in the following section). However, their

paper leaves open the computational complexity of finding the optimal Stackelberg strategy in their framework. In this chapter, we resolve the complexity of all the major variants in their framework, in some cases giving polynomial-time algorithms, and in other cases giving NP-hardness results.

4.1 Security Games: Problem Description and Notation

Following Kiekintveld et al. (2009), we consider the following two-player general-sum game. Player one (the "leader" or "defender") commits to a mixed strategy to allocate a set of resources to defend a set of targets.[1] Player two (the "follower" or "attacker") observes the commitment and then picks one of the targets to attack. The utilities of the players depend on which target was attacked and whether that target was defended.

We will consider several variants of this game. Resources of the leader can be either homogeneous, or there can be several types of resources, each with different limitations on what they can defend. It can either be the case that a resource can be assigned to at most one target, or it can be the case that a resource can be assigned to a subset of the targets (such a subset is called a *schedule*). As we will see, the complexity depends on the size of these schedules.

We will use the following notation to describe different variants of the problem.

- **Targets.** Described by a set T ($|T| = n$). A target t is *covered* if there is a resource assigned to t (in the case of no schedules), or if a resource is assigned to a schedule that includes t.

- **Schedules.** Described by a collection of subsets of targets $\Sigma \subseteq 2^T$. Here, $\sigma \in \Sigma$ is a subset of targets that can be simultaneously covered by some resource. We

[1] In this thesis, we assume that the set of resources is fixed, as is the case in practice in the short term. For long-term planning, it may be useful to consider settings where additional resources can be obtained at a cost (Bhattacharya et al., 2011; Letchford and Vorobeychik, 2012), but we will not do so in this thesis.

assume that any subset of a schedule is also a schedule, that is, if $\sigma' \subseteq \sigma$ and $\sigma \in \Sigma$, then $\sigma' \in \Sigma$. When resources are assigned to individual targets, we have (by a slight abuse of notation) $\Sigma = T \cup \{\varnothing\}$, where $\varnothing$ corresponds to not covering any target.

- **Resources.** Described by a set Ω ($|\Omega| = m$). When the resources are heterogeneous, a function $H : \Omega \to 2^{\Sigma}$ is given, where $H(\omega)$ is the set of schedules to which resource ω can be assigned. We assume that if $\sigma' \subseteq \sigma$ and $\sigma \in H(\omega)$, then $\sigma' \in H(\omega)$—that is, if a resource can cover a set of targets simultaneously, then it can also cover any subset of that set of targets simultaneously. If resources are homogeneous, then we assume every resource can cover all schedules, that is, $H(\omega) = \Sigma$ for all $\omega \in \Omega$.

- **Utility functions.** If target t is attacked, the defender's utility is $u_d^c(t)$ if t is covered, or $u_d^u(t)$ if t is not covered. The attacker's utility is $u_a^c(t)$ if t is covered, or $u_a^u(t)$ if t is not covered. We assume $u_d^c(t) \geqslant u_d^u(t)$ and $u_a^c(t) \leqslant u_a^u(t)$. We note that it makes no difference to the players' utilities whether a target is covered by one resource or by more than one resource.

LP notation. We will use linear programs in all of our positive results (polynomial-time algorithms). We now describe some of the variables used in these linear programs.

- c_t is the probability of target t being covered.

- c_s is the probability of schedule σ being covered.

- $c_{\omega,s}$ is the probability of resource ω being assigned to schedule σ.

Let $\mathbf{c}$ denote the vector of probabilities $(c_1, \ldots, c_n)$. Then, the utilities of the leader

and the follower can be computed as follows, given $\mathbf{c}$ and the target t being attacked:

$$u_d(t, \mathbf{c}) = c_t u_d^c(t) + (1 - c_t) u_d^u(t)$$

$$u_a(t, \mathbf{c}) = c_t u_a^c(t) + (1 - c_t) u_a^u(t)$$

These equalities are implicit in all of our linear programs and, for brevity, are not repeated.

4.2 Heterogeneous Resources, Singleton Schedules

We first consider the case in which schedules have size 1 or 0 (that is, resources are assigned to individual targets or not at all, so that $\Sigma = T \cup \{\varnothing\}$). We show that here, we can find an optimal strategy for the leader in polynomial time. Kiekintveld et al. (2009) gave a mixed-integer program formulation for this problem, and proved that feasible solutions for this program correspond to mixed strategies in the game. However, they did not show how to compute the mixed strategy in polynomial time. Our linear program formulation is similar to their formulation, and we show how to construct the mixed strategy from the solution, using the Birkhoff-von Neumann theorem (Birkhoff, 1946).

To solve the problem, we actually solve multiple LPs: for each target t^*, we solve an LP that computes the best mixed strategy to commit to, under the constraint that the attacker is incentivized to attack t^*. We then solve all of these LPs, and take the solution that maximizes the leader's utility. This is similar to the set of linear programs from Section 4.1, except those linear programs require a variable for each pure strategy for the defender, so that these LPs have exponential size in our domain. Instead, we will write a more compact LP to find the probability $c_{\omega,t}$ of assigning resource ω to target t, for each ω and $t \in H(\omega)$. (If $t \notin H(\omega)$, then there is

no variable $c_{\omega,t}$.)

$$\text{maximize } u_d(t^*, \mathbf{c})$$

$$\text{subject to}$$

$$\forall \omega \in \Omega, \forall t \in H(\omega) : 0 \leqslant c_{\omega,t} \leqslant 1$$

$$\forall t \in T : c_t = \sum_{\omega \in \Omega : t \in H(\omega)} c_{\omega,t} \leqslant 1$$

$$\forall \omega \in \Omega : \sum_{t \in H(\omega)} c_{\omega,t} \leqslant 1$$

$$\forall t \in T : u_a(t, \mathbf{c}) \leqslant u_a(t^*, \mathbf{c})$$

The advantage of this LP is that it is more compact than the one that considers all pure strategies. The downside is that it is not immediately clear whether we can actually implement the computed probabilities (that is, whether they correspond to a probability distribution over allocations of resources to targets, and how this mixed strategy can be found). Below we show that the obtained probabilities can, in fact, be implemented.

4.2.1 Constructing a Strategy that Implements the LP Solution

We will make heavy use of the following theorem which we state in a somewhat more general form than it is usually stated.

Theorem 3 (Birkhoff-von Neumann theorem, Birkhoff 1946)**.** *Consider an $m \times n$ matrix M with real numbers $a_{ij} \in [0, 1]$, such that for each $1 \leqslant i \leqslant m$, $\sum_{j=1}^{n} a_{ij} \leqslant 1$, and for each $1 \leqslant j \leqslant n$, $\sum_{i=1}^{m} a_{ij} \leqslant 1$. Then, there exist matrices $M^1, M^2, \ldots, M^q$, and weights $w^1, w^2, \ldots, w^q \in (0, 1]$, such that:*

1. $\sum_{k=1}^{q} w^k = 1$;
2. $\sum_{k=1}^{q} w^k M^k = M$;

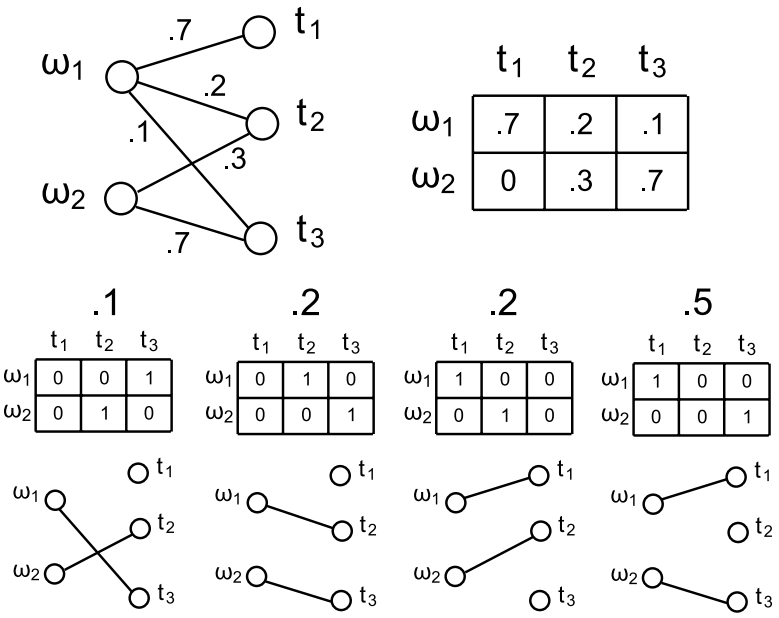

FIGURE 4.1: An example of how to apply the BvN theorem. Top Left: Resource ω_1 can cover targets t_1, t_2, t_3; ω_2 can cover t_2, t_3. Top Right: The LP returns the marginal probabilities in the table. We must now obtain these marginal probabilities as a probability mixture over pure strategies, in which every resource is assigned to a separate target. Bottom: The BvN theorem decomposes the top right table into a mixture over pure strategies. It first places probability .1 on the pure strategy on the left, then .2 on the pure strategy to the right of that, and so on. It is easily checked that with the resulting mixture over pure strategies, the marginal probabilities in the top right table are obtained.

3. *for each* $1 \leqslant k \leqslant q$, *the elements of* M^k *are* $a_{ij}^k \in \{0, 1\}$;

4. *for each* $1 \leqslant k \leqslant q$, *we have: for each* $1 \leqslant i \leqslant m$, $\sum_{j=1}^{n} a_{ij}^k \leqslant 1$, *and for each* $1 \leqslant j \leqslant n$, $\sum_{i=1}^{m} a_{ij}^k \leqslant 1$.

Moreover, q *is* $O((m+n)^2)$, *and the* M^k *and* w^k *can be found in* $O((m+n)^{4.5})$ *time using Dulmage-Halperin algorithm (Dulmage and Halperin, 1955; Chang et al., 2001).*

We can use this theorem to convert the probabilities $c_{\omega,t}$ that we obtain from our linear programming approach into a mixed strategy. This is because the $c_{\omega,t}$ constitute an $m \times n$ matrix that satisfies the conditions of the Birkhoff-von Neumann

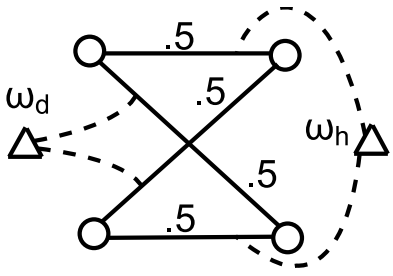

FIGURE 4.2: A counterexample that shows that with heterogeneous resources and bipartite schedules, the linear program probabilities are not always implementable. There are 4 targets (shown as circles), 4 schedules (solid edges), and 2 resources. The resource ω_h can be assigned to one of the horizontal edges and the resource ω_d can be assigned to one of the diagonal edges. In the optimal solution to the LP, the probability of a resource being assigned to each edge is 0.5, so that it would seem that the probability of each target being covered is 1. However, it is easy to see that in reality, the two resources can cover at most 3 of the 4 targets simultaneously.

theorem. Each M^k that we obtain as a result of this application of the theorem corresponds to a pure strategy in our domain: M^k consists of entries $c^k_{\omega,t} \in \{0,1\}$ (by 3), which we can interpret to mean that ω is assigned to t if and only if $c^k_{\omega,t} = 1$, because of the conditions on M^k (in 4). Then, because the weights sum to 1 (by 1), we can think of $\sum_{k=1}^{q} w^k M^k$ as a mixed strategy in our domain, which gives us the right probabilities (by 2). According to the theorem, we can construct this

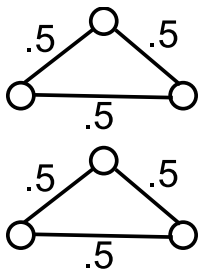

FIGURE 4.3: A counterexample that shows that with homogeneous resources and schedules of size two that are not bipartite, the linear program probabilities are not always implementable. The number of resources is $m = 3$. 6 targets are represented by vertices, 6 schedules are represented by edges. In the optimal solution to the LP, the probability of a resource being assigned to each edge is 0.5, so that it would seem that the probability of each target being covered is 1. However, it is easy to see that in reality, the three resources can cover at most 5 of the 6 targets simultaneously.

mixed strategy (represented as an explicit listing of the pure strategies in its support, together with their probabilities) in polynomial time. An example is shown on Figure 4.1. From this analysis, the following theorem follows:

Theorem 4. *When schedules have size 1 or 0, we can find an optimal Stackelberg strategy in polynomial time, even with heterogeneous resources. This can be done by solving a set of polynomial-sized linear programs and then applying the Birkhoff-von Neumann theorem.*

4.3 Heterogeneous Resources, Schedules of Size 2, Bipartite Graph

In this section, we consider schedules of size two. When schedules have size two, they can be represented as a graph, whose vertices correspond to targets and whose edges correspond to schedules. In this section, we consider the special case where this graph is bipartite, and give an NP-hardness proof for it.

One may wonder why this special case is interesting. In fact, it corresponds to the Federal Air Marshals domain studied by Kiekintveld et al. (2009). In this domain, flights are targets. If a Federal Air Marshal is to be scheduled on one outgoing flight from the U.S. (to, say, Europe), and will then return on an incoming flight, this is a schedule that involves two targets; moreover, there cannot be a schedule consisting of two outgoing flights or of two incoming flights, so that we have the requisite bipartite structure.

It may seem that the natural approach is to use a generalization of the linear program from the previous section (or, the mixed integer program by Kiekintveld et al. (2009)) to compute the marginal probabilities $c_{\omega,\sigma}$ that resource ω is assigned to schedule σ; and, subsequently, to convert this into a distribution over pure strategies that gives those marginal probabilities. However, it turns out that it is, in some cases, *impossible* to find such a distribution over pure strategies. That is, the

marginal probabilities from the linear program are not actually implementable. A counterexample is shown in Figure 4.2. One may wonder if perhaps a different linear program or other efficient algorithm can be given. We next show that this is unlikely, because finding an optimal strategy for the leader in this case is actually NP-hard, even in zero-sum games.

Theorem 5. *When resources are heterogeneous, finding an optimal Stackelberg strategy is NP-hard, even when schedules have size 2 and constitute a bipartite graph, and the game is zero-sum.*

Proof. We reduce an arbitrary satisfiability instance, given by variables V and clauses C, and reduce it to a game of the form in the theorem. Figure 4.4 illustrates the reduction. For each variable in V, we create a cyclic bipartite graph with $2|C|$ vertices (where the vertices are targets and the edges are schedules). Also, for each clause in C, we create another two vertices and an edge between these two vertices. Finally, we create $|C| \cdot (|V| - 1)$ additional "dummy" schedules, each consisting of two targets with an edge between them. For each variable x_i, we create $2|C|$ resources, $\omega^1_{+x_i}, \ldots, \omega^{|C|}_{+x_i}$ (corresponding to $+x_1$—these are $|C|$ homogeneous resources) and $\omega^1_{-x_i}, \ldots, \omega^{|C|}_{-x_i}$ (corresponding to $-x_1$—again, these are $|C|$ homogeneous resources). Resource $\omega_{+x_i^k}$ (for any k) can be assigned to:

- any even (horizontal) edge in the cyclic bipartite graph corresponding to variable x_i;
- any edge corresponding to a clause that includes $+x_i$; and
- any dummy edge.

Similarly, resource $\omega_{-x_i^k}$ (for any k) can be assigned to:

- any odd (diagonal) edge in the cyclic bipartite graph corresponding to variable x_i;

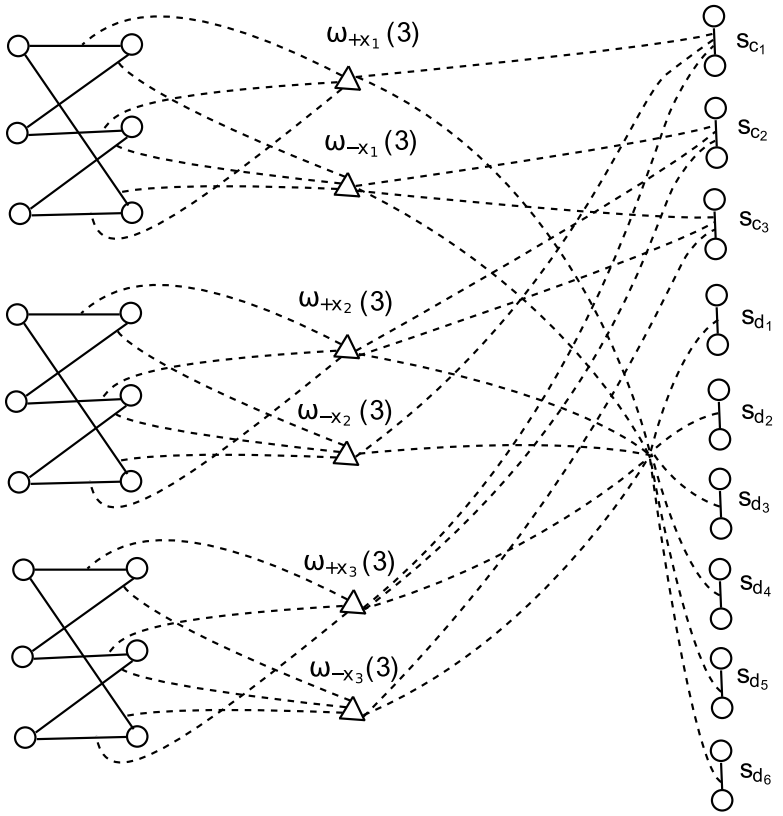

FIGURE 4.4: Illustration for the NP-hardness reduction for heterogeneous resources, schedules of size 2 constituting a bipartite graph. The boolean formula is $(+x_1 \vee -x_2 \vee +x_3) \wedge (-x_1 \vee +x_2 \vee +x_3) \wedge (-x_1 \vee +x_2 \vee -x_3)$. There are 36 targets (shown as circles), 27 schedules (shown as solid edges), and 6 types of resources (shown as triangles), of which there are 3 copies in each case. A dashed line from a resource type to a schedule indicates that that resource can be assigned to that schedule. The three cyclic bipartite graphs each correspond to a variable x_i; the schedules σ_{c_i} correspond to the three clauses in the formula.

- any edge corresponding to a clause that includes $-x_i$; and
- any dummy edge.

Finally, let the utilities be: $\forall t : u_d^c(t) = 1, u_d^u(t) = 0, u_a^c(t) = 0, u_a^u(t) = 1$, so that we have a zero-sum game.

We now prove that the optimal utility for the leader in this game is 1 if and only if the corresponding boolean formula is satisfiable.

The "if" case: Suppose there is an assignment of values $\{0, 1\}$ to the variables such that the formula evaluates to 1. Then we can assign the resources in the game in the following way.

- For every clause in the boolean formula, choose one literal ($+x_i$ or $-x_i$) that evaluates to "true" in the assignment. Assign one of the corresponding resources ($\omega^k_{+x_i}$ or $\omega^k_{-x_i}$, for some k) to the schedule corresponding to the clause.
- For every $x_i = 0$ ($x_i = 1$), we assign all resources $\omega^k_{+x_i}$ ($\omega^k_{-x_i}$) to cover the entire cyclic bipartite graph corresponding to x_i, using the horizontal (diagonal) edges.
- Assign the remaining resources to cover all the "dummy" schedules.

The resulting assignment of resources covers all targets, so that if the defender plays this pure strategy, the defender's utility is 1 in the security game.

The "only if" case: Suppose there is a defender strategy σ that gives the defender a utility of 1 in the Stackelberg game. Then, every target must be covered with probability 1. That means that any pure strategy on which σ puts positive probability must cover all the targets. So, without loss of generality, we can assume σ is a pure strategy. We note that there are $2|C| \cdot |V|$ resources and $4|C| \cdot |V|$ targets in the game. Since each schedule covers 2 targets, in σ, no two resources are assigned to schedules that share a target. Thus, each cyclic bipartite graph corresponding to a variable x_i must have either all its horizontal edges covered (using all the $\omega^k_{+x_i}$), or all its diagonal edges covered (using all the $\omega^k_{-x_i}$). If all the $\omega^k_{+x_i}$ are used to cover horizontal edges, we set x_i to 1; if all the $\omega^k_{-x_i}$ are used to cover diagonal edges, we set x_i to 0. Now, for every clause $c \in C$, the corresponding schedule is covered by some resource. If it is covered by some $\omega^k_{+x_i}$, then $+x_i \in c$, and because in that case all the $\omega^k_{-x_i}$ must be used to cover the cyclic bipartite graph corresponding to

x_i (using the diagonal edges), it must be that $x_i = 1$. Similarly, if it is covered by some $\omega^k_{-x_i}$, then $-x_i \in c$, and it must be that $x_i = 0$. Hence we have a satisfying assignment. □

If the resources are homogeneous, then it turns out that in the bipartite case, we can solve for an optimal Stackelberg strategy in polynomial time, by using the Birkhoff-von Neumann theorem in a slightly different way. We consider that case in the following section.

4.4 Homogeneous Resources, Schedules of Size at Most 2, Bipartite Graph

In this section, we restrict ourselves to homogeneous resources, but now we consider schedules of size two. When schedules have size two, they can be represented as a graph, whose vertices correspond to targets and whose edges correspond to schedules. In this section, we consider the special case where this graph is bipartite, and give a polynomial-time solution for it, again based on the Birkhoff-von Neumann theorem.

One may wonder why this special case is interesting. In fact, it corresponds to the Federal Air Marshals domain studied by Kiekintveld et al. (2009). In this domain, flights are targets. If a Federal Air Marshal is to be scheduled on one outgoing flight from the U.S. (to, say, Europe), and will then return on an incoming flight, this is a schedule that involves two targets; moreover, there cannot be a schedule consisting of two outgoing flights or of two incoming flights, so that we have the requisite bipartite structure.

We will use the following linear program to compute, for each schedule, the probability that one of the (homogeneous) resources is assigned to that schedule.

(Again, we need to solve n of these linear programs, one for each value of t^*.)

$$\text{maximize } u_d(t^*, \mathbf{c})$$

$$\text{subject to}$$

$$\forall \sigma \in \Sigma : c_\sigma \leqslant 1$$

$$\forall t \in T : c_t = \sum_{\sigma \in \Sigma : t \in \sigma} c_\sigma \leqslant 1$$

$$\sum_{\sigma \in \Sigma} c_\sigma \leqslant m$$

$$\forall t \in T : u_a(t, \mathbf{c}) \leqslant u_a(t^*, \mathbf{c})$$

We note that nothing about this program specifically corresponds to the bipartite-schedules-of-size-two case. Indeed, it is very similar to the mixed integer program for the general case presented by Kiekintveld et al. (2009). However, in the general case, the solutions returned by both this linear program and the known mixed integer program do *not* always correspond to implementable mixed strategies (we will give counterexamples shortly). The contribution of this section is to show that in the bipartite-schedules-of-size-two case, the solutions are in fact implementable.

4.4.1 Constructing a Strategy that Implements the LP Solution

Again, we will use the Birkhoff-von Neumann theorem to show this. Actually, we will use a slightly stronger version of the theorem (which is not difficult to prove using the basic version, Theorem 3), as follows:

Theorem 6 (Strengthening of BvN)**.** *If the matrix M in Theorem 3 additionally satisfies $\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} \leqslant p$, where p is an integer, then we can obtain matrices M^k that additionally have the property that for each $1 \leqslant k \leqslant q$, $\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^k \leqslant p$.*

Again, we will construct a matrix of probabilities M, where the probabilities in each row and the probabilities in each column sum to at most 1. Because the graph

is bipartite, the targets partition into T_1, T_2, with no edges inside T_1 or inside T_2. The substochastic matrix M is constructed as follows.

$$M = \begin{pmatrix} M_s & M_{T_1} \\ M_{T_2} & 0_{|T_2|\times|T_1|} \end{pmatrix}$$

Here, $0_{|T_2|\times|T_1|}$ is a matrix of size $|T_2| \times |T_1|$, consisting of only zeroes. The rows of the submatrix M_s correspond to the targets in T_1, and the columns of M_s correspond to the targets in T_2. Each entry of the matrix M_s is the probability $c_{\{t_1,t_2\}}$ that the schedule $\{t_1, t_2\}$ is covered.

To represent the schedules of size 1, we use two square diagonal submatrices M_{T_1} and M_{T_2}. For $y \in \{1, 2\}$, element (i, i) of matrix M_{T_y} is equal to the probability $c_{\{t_i\}}$ of a resource being assigned to cover target $t_i \in T_y$ by itself; all the entries off the diagonal are 0.

By applying the strengthened Birkhoff-von Neumann theorem, we can decompose M into a convex combination of 0-1 matrices M^k, such that $M = \sum_{k=1}^{q} w^k M^k$. Every row and every column of each matrix M^k contains no more than one element equal to 1, and the total number of 1s in M^k is no more than m. Each M^k corresponds to a pure strategy for the defender. The pure strategy corresponding to 0-1 matrix M^k is to place a defensive resource on every schedule of size two for which the corresponding cell of the submatrix M_s is equal to 1, and to place a resource on every schedule of size one for which the corresponding cell of the diagonal submatrix M_{T_1} or M_{T_2} is equal to 1. No target is covered twice in any one of these pure strategies, because for each $t \in T_1$ ($t \in T_2$), there is at most one 1 in the row (column) corresponding to that t. The mixed strategy is to play the pure strategy corresponding to M^k with probability w^k, for $1 \leqslant k \leqslant q$. By doing so, the probability of covering target $t \in T_1$ is indeed $c_{\{t\}} + \sum_{t \in T_2} c_{\{t_1,t_2\}} = c_t$ (and similarly for $t \in T_2$). Hence, the marginal probabilities obtained from the linear program correspond to an implementable strategy.

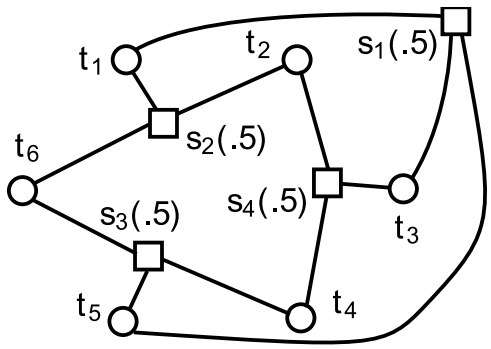

FIGURE 4.5: A counterexample that shows that with homogeneous resources and schedules of size three, the linear program probabilities are not always implementable. The number of resources is $m = 2$. 6 targets are represented by round nodes, 6 schedules are represented by square nodes with connections to the targets that they include. In the optimal solution to the LP, the probability of a resource being assigned to each schedule is 0.5, so that it would seem that the probability of each target being covered is 1. However, it is easy to see that in reality, the two resources can cover at most 5 of the 6 targets simultaneously.

Theorem 7. *When resources are homogeneous and schedules have size at most 2, and constitute a bipartite graph, we can find an optimal Stackelberg strategy in polynomial time. This can be done by solving a set of polynomial-sized linear programs and then applying the Birkhoff-von Neumann theorem.*

4.5 Homogeneous Resources, Schedules of Size 2

We now return to the case where resources are homogeneous and schedules have size 2, but now we no longer assume that the graph is bipartite. It turns out that if we use the linear program approach, the resulting marginal probabilities c_s are in general not implementable, that is, there is no mixed strategy that attains these marginal probabilities. A counterexample is shown in Figure 4.3. This would appear to put us in a position similar to that in the previous section. However, it turns out that here we can actually solve the problem in polynomial time, using a different approach. Our approach here is to use the standard linear programming approach from Section 4.1. The downside of using such approach is that there are exponentially

many variables. In contrast, the dual linear program has only $n + 1$ variables, but exponentially many constraints. One approach to solving a linear program with exponentially many constraints is the following: start with only a small subset of the constraints, and solve the resulting reduced linear program. Then, using some other method, check whether the solution is feasible for the full (original) linear program; and if not, find a violated constraint. If we have a violated constraint, we add it to the set of constraints, and repeat. Otherwise, we have found an optimal solution. This process is known as *constraint generation*. Moreover, if a violated constraint can be found in polynomial time, then the original linear program can be solved in polynomial time using the ellipsoid algorithm. As we will show, in the case of homogeneous resources and schedules of size two, we can efficiently generate constraints in the dual linear program by solving a weighted matching problem. While this solution is less appealing than our earlier solutions based on the Birkhoff-von Neumann theorem, it still results in a polynomial-time algorithm. The dual linear program follows.

$$\text{minimize } y$$

$$\text{subject to}$$

$$\forall \alpha : \sum_{t \in T} y_t(u_a(\alpha, t) - u_a(\alpha, t^*)) + y \geqslant u_d(\alpha, t^*)$$

$$y \in \mathbb{R}$$

Now, we consider the constraint generation problem for the dual LP. Given a (not necessarily feasible) solution y_t, y to the dual, we need to find the most violated constraint, or verify that the solution is in fact feasible. Our goal is to find, given the candidate solution y_t, y,

$$\alpha \in \arg\max_{\alpha} u_d(\alpha, t^*) - \sum_{t \in T} y_t(u_a(\alpha, t) - u_a(\alpha, t^*)) - y$$

We introduce an indicator function $I_\alpha(t)$ which is equal to 1 if t is covered by α, and 0 otherwise. Then

$$u_a(\alpha, t) = u_a^u(t) + I_\alpha(t)(u_a^c(t) - u_a^u(t))$$

$$u_d(\alpha, t) = u_d^u(t) + I_\alpha(t)(u_d^c(t) - u_d^u(t))$$

Then, we can rearrange the optimization problem as follows.

$$\begin{aligned}
\alpha \in \arg\max_\alpha & \, u_d^u(t^*) + I_\alpha(t^*)(u_d^c(t^*) - u_d^u(t^*)) \\
& - y - \sum_{t \in T} y_t (u_a^u(t) + I_\alpha(t)(u_a^c(t) - u_a^u(t))) \\
& + \sum_{t \in T} y_t (u_a^u(t^*) + I_\alpha(t^*)(u_a^c(t^*) - u_a^u(t^*))) \\
= & \, u_d^u(t^*) - y - \sum_{t \in T} y_t (u_a^u(t) - u_a^u(t^*)) \\
& + \sum_{t \in T} I_\alpha(t) y_t (u_a^u(t) - u_a^u(t)) \\
& - I_\alpha(t^*)(u_a^u(t^*) - u_a^c(t^*)) \sum_{t \in T} y_t
\end{aligned}$$

We define a weight function on the targets as follows:

$$w(t) = y_t(u_a^u(t) - u_a^c(t)) \quad \text{for } t \neq t^*$$

$$w(t^*) = -(u_a^u(t^*) - u_a^c(t^*)) \sum_{t \in T, t \neq t^*} y_t$$

We then rearrange the optimization problem as follows:

$$\alpha \in \arg\max_\alpha w(\alpha) + u_d^u(t^*) - \sum_{t \in T} y_t (u_a^u(t) - u_a^u(t^*)) - y$$

where $w(\alpha)$ is the total weight of the targets covered by the pure strategy α: $w(\alpha) = \sum_{t \in \cup_{s \in \alpha} s} w(t)$. The only part of the objective that depends on α is $w(\alpha)$, so we can focus on finding an α that maximizes $w(\alpha)$. A pure strategy α is a collection of

edges (schedules consisting of pairs of targets). Therefore, the problem of finding an α with maximum weight is a maximum weighted 2-cover problem, which can be solved in polynomial time (for example, using a modification of the algorithm for finding a maximal weighted matching in general graphs (Galil et al., 1986)). So, we can solve the constraint generation problem, and hence the whole problem, in polynomial time. From this analysis, the following theorem follows:

Theorem 8. *When resources are homogeneous and schedules have size at most 2, we can find an optimal Stackelberg strategy in polynomial time. This can be done by solving the standard Stackelberg linear programs (Conitzer and Sandholm, 2006; von Stengel and Zamir, 2010): these programs have exponentially many variables, but the constraint generation problem for the dual can be solved in polynomial time in this case.*

4.6 Homogeneous Resources, Schedules of Size 3

We now move on to the case of homogeneous resources with schedules of size 3. Once again, it turns out that if we use the linear program approach, the resulting marginal probabilities c_s are in general not implementable; that is, there is no mixed strategy that attains these marginal probabilities. A counterexample is shown in Figure 4.5.

We now show that finding an optimal strategy for the leader in this case is actually NP-hard, even in zero-sum games.

Theorem 9. *When schedules have size 3, finding an optimal Stackelberg strategy is NP-hard, even when resources are homogeneous and the game is zero-sum.*

Proof. We reduce an arbitrary 3-cover problem instance—in which we are given a universe $\mathcal{U}$, a family $\mathcal{S}$ of subsets of $\mathcal{U}$, such that each subset contains 3 elements, and we are asked whether we can (exactly) cover all of $\mathcal{U}$ using $|\mathcal{U}|/3$ elements of $\mathcal{S}$—to a game with homogeneous resources and schedules of size 3. We create one

target for each element of $\mathcal{U}$, and one schedule for each element of $\mathcal{S}$, which covers the targets in it. We also create $|\mathcal{U}|/3$ homogeneous resources that each can cover any schedule. The utilities are $\forall t : u_d^c(t) = u_a^u(t) = 1, u_d^u(t) = u_a^c(t) = 0$.

First, we will show that the answer is "yes" in the set cover problem instance if there is defender strategy that gives the defender the utility of 1. Suppose there is defender strategy C that gives the defender the utility of 1 in the Stackelberg setting. Since the game is zero sum, the attacker's utility must be 0. It must be that $u_a(t, C) = 0$ for each $t \in T$. That implies $c_t = 1$ for all t. Then each target must be covered in each pure strategy over which the defender is mixing in C. Thus there exists a pure strategy that covers each target with no more than k resources. The set of schedules covered in that strategy corresponds to the subfamily $\mathcal{C}$ of size no more than k such that the union of subsets in $\mathcal{C}$ is exactly $\mathcal{U}$.

Next, we will show that if there exists a subfamily $\mathcal{C}$ of size no more than k, then there is defender strategy that gives the defender utility of 1 in the Stackelberg game. Consider a pure strategy in which the defender assigns a resource to each schedule that corresponds to a subset in $\mathcal{C}$. Such pure strategy covers every target. If the defender plays this pure strategy, attacker's utility from attacking any target is equal to 0. Since the game is zero sum, the defender's utility is 1.

We have shown that the answer to the 3-Set Cover problem is positive if and only if the leader's utility in the corresponding resource allocation game is 1. □

4.7 Discussion

In this chapter, we characterized in which security games the problem of computing a Stackelberg strategy is solvable in polynomial time and in which cases it is NP-hard. We summarize thre results in Figure 4.7.

Our results are perhaps made more interesting by the paper by Korzhyk et al. (2011), which shows that for all of the security games that we studied, an optimal

	Schedules			
Heterogeneous resources	**size 1**	**size $\leqslant$ 2, bipartite**	**size $\leqslant$ 2**	**size $\geqslant$ 3**
No	P	P	P	NP-hard
Yes	P	NP-hard	NP-hard	NP-hard

FIGURE 4.6: Summary of the computational results. All of the hardness results hold even for zero-sum games.

Stackelberg strategy is guaranteed to also be a Nash equilibrium strategy in the version of the game where commitment is not possible. (The converse does not hold, that is, there can be Nash equilibrium strategies that are not Stackelberg strategies.) Thus, our polynomial-time algorithm results also allow us to find a Nash equilibrium strategy for the defender in polynomial time. Conversely, for the cases where we prove a hardness result, finding a Nash equilibrium strategy is also NP-hard, because our hardness results hold even for zero-sum games.

In the next chapter, we consider an extended security games model in which the attacker may choose multiple targets for a simultaneous attack.

5

Security Games with Multiple Attacker Resources

The security games model by Kiekintveld et al. (2009) discussed in the previous chapter was further extended by Korzhyk et al. (2011) to allow multiple attacker resources; that is, the attacker can simultaneously attack up to n_a different targets. This extension is motivated by the fact that terrorist attacks are often performed at multiple locations simultaneously (for example, the 9/11 attacks or the 2008 Mumbai attacks).

In the case of a single attacker resource, Kiekintveld et al. give a simple algorithm called ORIGAMI for the single attacker resource case. It computes a defender strategy that is both a Stackelberg and a Nash strategy (the latter follows from the work of Korzhyk et al. (2011)). The main observation used in ORIGAMI is that any Stackelberg strategy for the defender minimizes the attacker's best-response utility. Using this observation, ORIGAMI computes the defender's Nash/Stackelberg strategy by gradually decreasing the attacker's best-response utility, keeping track of the number of defender resources required to bring the attacker's best-response utility down to this level, until the number of required defender resources reaches the limit.

However, in games with multiple attacker resources, the defender's minimax strategy is not necessarily a Nash or Stackelberg strategy. Consider the following example (by Korzhyk et al. (2011)). Suppose there are two targets, and the attacker has two resources, so that both targets will be attacked no matter what strategy the defender chooses. If the defender has only one resource, then the defender is better off allocating that resource in such a way that the defender's utility increases the most. However, in the defender's minimax strategy, the defender would allocate the resource so that the attacker's utility is reduced the most. Thus the defender's Nash and Stackelberg strategies can differ from the minimax strategy in this example.

In this chapter, we give a polynomial-time algorithm for finding a Nash equilibrium in the case of multiple attacker resources. This algorithm can be thought of as a generalization of ORIGAMI in the sense that it also keeps track of the smallest utility that the attacker is going to get from any of his targets, and this utility gradually decreases over the course of the algorithm. However, our algorithm is far more complicated compared to ORIGAMI. Furthermore, we show that Nash equilibria in security games with multiple attacker resources possess the interchange property, which states that as long as each player plays *some* equilibrium strategy, the resulting strategy profile must be a Nash equilibrium, thus resolving the problem of equilibrium selection for both players. On the other hand, we show that, with multiple attacker resources, computing a defender Stackelberg strategy is actually NP-hard.

5.1 The Model

In the security games that we study, there is a set of *targets*, T. The defender has n_d resources and the attacker has n_a resources. A pure strategy for the defender (attacker) consists of a subset $a_d \subseteq T$ with $|a_d| = n_d$ ($a_a \subseteq T$ with $|a_a| = n_a$), corresponding to the targets she defends (he attacks). Targets that are not attacked do not affect either player's utility. Each player's utility is additive over attacked

targets. For a target t, the defender receives utility $u_d^u(t)$ if the target is attacked but not defended (uncovered), and $u_d^c(t)$ if the target is attacked and defended (covered). Similarly, the attacker's utility for an attacked target t is $u_a^u(t)$ in the uncovered case and $u_a^c(t)$ in the covered case. (We require $\Delta u_d(t) = u_d^c(t) - u_d^u(t) > 0$ and $\Delta u_a(t) = u_a^u(t) - u_a^c(t) > 0$.) Hence, the defender's overall utility is $\sum_{t \in a_a \cap a_d} u_d^c(t) + \sum_{t \in a_a \setminus a_d} u_d^u(t)$, and the attacker's overall utility is $\sum_{t \in a_a \cap a_d} u_a^c(t) + \sum_{t \in a_a \setminus a_d} u_a^u(t)$.

Because of the additive nature of the utility function, the players' expected utilities depend only on the marginal probability that each target is attacked/defended. Hence, a defender mixed strategy can be represented as a vector $s_d = (s_d(t_1), \ldots, s_d(t_{|T|}))$ with $\sum_{t \in T} s_d(t) = n_d$, where $s_d(t) \in [0, 1]$ is the probability that target t obtains a defensive resource, and similarly for the attacker $s_a = (s_a(t_1), \ldots, s_a(t_{|T|}))$ with $\sum_{t \in T} s_a(t) = n_a, s_a(t) \in [0, 1]$. (Note that it does not help to have more than one resource on one target. This assumption was introduced in the security game model by Kiekintveld et al. (2009).)

We will use the following shorthand: $u_a(t, s_d(t)) = s_d(t)u_a^c(t) + (1 - s_d(t))u_a^u(t)$ is the attacker's expected utility for attacking target t, and $\upsilon_d(t, s_a(t)) = s_a(t)\Delta u_d(t)$ is the marginal expected utility that the defender gets from defending t.

5.2 Nash Equilibrium

We now consider how to compute a Nash equilibrium of a security game with multiple attacker resources. Because of the additive nature of the utility functions, best-responding simply means defending (attacking) the n_d (n_a) targets with the highest utility for the defender (attacker). If there is a tie for the n_dth (n_ath) place, then it is possible to randomize over the corresponding targets. Therefore, s_d is a best response to s_a iff there exists some threshold marginal utility θ_d such that $\upsilon_d(t, s_a(t)) < \theta_d \Rightarrow s_d(t) = 0$ and $\upsilon_d(t, s_a(t)) > \theta_d \Rightarrow s_d(t) = 1$. Similarly, s_a is a best response to s_d iff there exists some threshold utility θ_a such that $u_a(t, s_d(t)) < \theta_a \Rightarrow s_a(t) = 0$ and

$u_a(t, s_d(t)) > \theta_a \Rightarrow s_a(t) = 1$. Hence, we have a Nash equilibrium iff both these conditions are satisfied. Note that the defender's (and, similarly, the attacker's) strategy can be mixed (randomized), because any target t such that $\upsilon_d(t, s_a(t)) = \theta_d$ can be defended with a fractional probability $s_d(t) \in [0, 1]$. Similarly, the attacker's strategy can also be mixed.

The high-level idea of our algorithm for computing a Nash equilibrium is as follows. We start with a modified game where the defender has no resources at all, for which it is straightforward to compute an equilibrium, and then we gradually increase the number of defender resources (not necessarily to integral amounts), while maintaining an equilibrium of the game as it is changing—until we arrive at the actual number of defender resources. The algorithm transitions among phases that correspond to phases of qualitatively different behavior in the process of increasing the number of defender resources. The change resulting from a single phase can be computed through a simple calculation.

5.2.1 Detailed Example Run of the Algorithm

Before we present the algorithm for computing a Nash equilibrium, we first give a detailed example of how it solves a particular game. This example demonstrates the different phases of the algorithm. During these phases, each target will be considered to be in one of 6 possible states: Untouched (U), Newly Attacked (NA), Pending (P), Active (A), Defender Saturated (DS), and Done (D). We will informally introduce every state in this example (precise definitions can be found in the algorithm in Figure 5.2). The target states depend on the values of the thresholds θ_a, θ_d, which are computed at the beginning and gradually decrease during the course of the algorithm.

Example 1. *Consider a game with $|T| = 4$ targets, $n_d = 3$ defender resources, and $n_a = 2$ attacker resources. The utilities are as follows:*

	t_1[P]	t_2[P]	t_3[NA]	t_4[U]
d_{ti}	0	0	0	0
a_{ti}	1	1	0	0
$u_a(t_i,\mathbf{c})$	5	4	3	2
$v_d(t_i,\mathbf{a})$	1	2	0	0

(a) Initialize. θ_d=2, θ_a=3

	t_1[P]	t_2[A]	t_3[NA]	t_4[U]
d_{ti}	0	**1/3**	0	0
a_{ti}	1	1	0	0
$u_a(t_i,\mathbf{c})$	5	**3**	3	2
$v_d(t_i,\mathbf{a})$	1	2	0	0

(b) IMOP. θ_d=2, θ_a=3

	t_1[P]	t_2[A]	t_3[A]	t_4[U]
d_{ti}	0	1/3	0	0
a_{ti}	1	**3/5**	**2/5**	0
$u_a(t_i,\mathbf{c})$	5	3	3	2
$v_d(t_i,\mathbf{a})$	1	**6/5**	**6/5**	0

(c) MMNA. θ_d=6/5, θ_a=3

	t_1[P]	t_2[A]	t_3[A]	t_4[NA]
d_{ti}	0	**2/3**	**1/3**	0
a_{ti}	1	3/5	2/5	0
$u_a(t_i,\mathbf{c})$	5	**2**	**2**	2
$v_d(t_i,\mathbf{a})$	1	6/5	6/5	0

(d) IMOA. θ_d=6/5, θ_a=2

	t_1[P]	t_2[A]	t_3[A]	t_4[NA]
d_{ti}	0	2/3	1/3	0
a_{ti}	1	**1/2**	**1/3**	**1/6**
$u_a(t_i,\mathbf{c})$	5	2	2	2
$v_d(t_i,\mathbf{a})$	1	**1**	**1**	**1/12**

(e) MMNA. θ_d=1, θ_a=2

	t_1[A]	t_2[A]	t_3[A]	t_4[NA]
d_{ti}	**3/5**	2/3	1/3	0
a_{ti}	1	1/2	1/3	1/6
$u_a(t_i,\mathbf{c})$	**2**	2	2	2
$v_d(t_i,\mathbf{a})$	1	1	1	1/12

(f) IMOP. θ_d=1, θ_a=2

	t_1[A]	t_2[A]	t_3[A]	t_4[P]
d_{ti}	3/5	2/3	1/3	0
a_{ti}	**6/11**	**3/11**	**2/11**	**1**
$u_a(t_i,\mathbf{c})$	2	2	2	2
$v_d(t_i,\mathbf{a})$	**6/11**	**6/11**	**6/11**	**1/2**

(g) MMNA. θ_d=6/11, θ_a=2

	t_1[A]	t_2[DS]	t_3[A]	t_4[P]
d_{ti}	**4/5**	**1**	**2/3**	0
a_{ti}	6/11	3/11	2/11	1
$u_a(t_i,\mathbf{c})$	1	1	1	2
$v_d(t_i,\mathbf{a})$	**6/11**	**6/11**	**6/11**	1/2

(h) IMOA. θ_d=6/11, θ_a=1

	t_1[A]	t_2[DS]	t_3[A]	t_4[P]
d_{ti}	4/5	1	2/3	0
a_{ti}	**1/2**	**1/3**	**1/6**	1
$u_a(t_i,\mathbf{c})$	1	1	1	2
$v_d(t_i,\mathbf{a})$	**1/2**	**2/3**	**1/2**	1/2

(i) MMDS. θ_d=1/2, θ_a=1

	t_1[A]	t_2[DS]	t_3[A]	t_4[A]
d_{ti}	4/5	1	2/3	**1/2**
a_{ti}	1/2	1/3	1/6	1
$u_a(t_i,\mathbf{c})$	1	1	1	**1**
$v_d(t_i,\mathbf{a})$	1/2	2/3	1/2	1/2

(j) IMOP. θ_d=1/2, θ_a=1

	t_1[A]	t_2[D]	t_3[A]	t_4[A]
d_{ti}	4/5	1	2/3	1/2
a_{ti}	**3/10**	**1**	**1/10**	**3/5**
$u_a(t_i,\mathbf{c})$	1	1	1	1
$v_d(t_i,\mathbf{a})$	**3/10**	**2**	**3/10**	**3/10**

(k) MMDS. θ_d=3/10, θ_a=1

	t_1	t_2	t_3	t_4
d_{ti}	**≈.806**	1	**≈.677**	**≈.516**
a_{ti}	3/10	1	1/10	3/5
$u_a(t_i,\mathbf{c})$	**≈.968**	1	**≈.968**	**≈.968**
$v_d(t_i,\mathbf{a})$	3/10	2	3/10	3/10

(l) IMOA. θ_d=3/10, θ_a≈.968

FIGURE 5.1: The example run of the algorithm. Each subfigure shows the current equilibrium, threshold values, and target states *at the end* of the corresponding phase.

	t_1	t_2	t_3	t_4
u_a^u	5	4	3	2
u_a^c	0	1	0	0
Δu_d	1	2	3	.5

Figure 5.1 shows the sequence of equilibria for different amounts of defender resources computed by the algorithm. We start with the equilibrium for 0 *defender resources. The attacker attacks the two targets that give him the highest utility, namely,* t_1 *and* t_2 *(Fig. 5.1(a)). Since these two targets are attacked with probability* 1*, they are likely to get defended as the number of defender resources increases. Thus, these targets are both in the Pending (P) state. Target* t_4 *is in the Untouched (U) state, because* $u_a(t_4) < \theta_a$*, and thus it is neither attacked nor defended.*

Increase Defender Mass on Pending (IMOP) phase: We increase the number of defender resources and allocate them to the Pending target t_2*, which is currently the most appealing to the defender. We cannot put more than* $1/3$ *defender probability on* t_2 *without breaking the attacker's equilibrium condition, so the phase ends at that point. The new equilibrium strategies and updated target states are shown in Figure 5.1(b).*

Move Attacker Mass to Newly Attacked (MMNA) phase: At the beginning of this phase, target t_2 is in the Active (A) state, because both players' utilities for defending/attacking this targets are at the corresponding threshold values. Target t_3 is in the Newly Attacked (NA) state, because it is currently at the attacker threshold and would start to be attacked if the defender put more probability on t_2. In this phase, we move $2/5$ of the attacker's probability mass from t_2 to t_3. As a result, both t_2 and t_3 now have the highest marginal defender utility (Fig. 5.1(c)).

Increase Defender Mass on Active (IMOA) phase: We increase the defender probability on Active targets t_2 and t_3. Since $\Delta u_a(t_2) = \Delta u_a(t_3)$, we have to add the same amount of probability to each of these targets; otherwise, the attacker's best-response condition would be broken. We can add up to $1/3$ defender probability to these targets, until the attacker's utility for attacking t_4 becomes equal to the utility for attacking t_2 and t_3 (Fig. 5.1(d)).

MMNA phase: We now move attacker mass from Active targets t_2 and t_3 to the Newly Attacked t_4. To maintain the defender's best-response condition, we need to take mass from t_2 and t_3 proportionally to $1/\Delta u_d(t)$. We stop when the defender becomes indifferent between t_2, t_3, and the Pending target t_1 (Fig. 5.1(e)).

IMOP phase: We add $3/5$ defender mass to t_1, after which the attacker becomes indifferent between all targets (Fig. 5.1(f)).

MMNA phase: The defender cannot add any mass to her optimal targets t_1, t_2, and t_3, because that would make t_4 strictly preferred for the attacker, and the attacker's best-response condition would be broken. Therefore, we move attacker mass from t_1, t_2, t_3 to t_4 in the right proportions, until the probability on t_4 reaches 1 (Fig. 5.1(g)).

IMOA phase: We can now add defender mass to t_1, t_2, t_3. That will make t_4 strictly preferred for the attacker. However, as long as we add mass in the right proportions, the attacker will still be best-responding, because t_4 is attacked with probabil-

MainRoutine

Initialize

`Repeat:`

`If` (no defender mass is left), `return` *UpdateTargetStates*

`If` (for all t, $a_t > 0$ implies $d_t = 1$), distribute the remaining defender resources arbitrarily and `return`

`Else if` (exists t in P s.t. $v_d(t, a_t) = \theta_d$), *IMOP*

`Else if` ($A \neq \emptyset$, $NA = \emptyset$, $DS = \emptyset$), *IMOA*

`Else if` ($A \neq \emptyset$, $NA \neq \emptyset$), *MMNA*

`Else if` ($A \neq \emptyset$, $DS \neq \emptyset$), *MMDS*

`Else` *DDT*

Initialize

Order the targets by decreasing $u_a^u(t)$, breaking ties arbitrarily

$a_t \leftarrow 1$ for the first n_a targets; $a_t \leftarrow 0$ for the rest

$\theta_a \leftarrow u_a^u(t^{n_a+1})$, where t^{n_a+1} is the $(n_a+1)^{st}$ target

$\theta_d \leftarrow \max_t a_t \Delta u_d(t, a_t)$

UpdateTargetStates

$U \leftarrow \{t: u_a(t, d_t) < \theta_a, a_t = 0, v_d(t, a_t) \leq \theta_d, d_t = 0\}$

$A \leftarrow \{t: u_a(t, d_t) = \theta_a, v_d(t, a_t) = \theta_d, d_t < 1\}$

$P \leftarrow \{t: u_a(t, d_t) > \theta_a, a_t = 1, v_d(t, a_t) = \theta_d, d_t < 1\}$
$\cup \{t: u_a(t, d_t) \geq \theta_a, a_t = 1, v_d(t, a_t) < \theta_d, d_t = 0\}$

$NA \leftarrow \{t: u_a(t, d_t) = \theta_a, a_t < 1, v_d(t, a_t) < \theta_d, d_t = 0\}$

$DS \leftarrow \{t: v_d(t, a_t) \geq \theta_d, d_t = 1, u_a(t, d_t) = \theta_a, a_t < 1\}$

$D \leftarrow \{t: v_d(t, a_t) \geq \theta_d, d_t = 1, u_a(t, d_t) \geq \theta_a, a_t = 1\}$

IMOP (Increase Defender Mass on Pending)

Choose $t^* \in P$ s.t. $v_d(t^*, a_t) = \theta_d$

Increase d_{t^*} until one of the following:

(1) $u_a(t^*, d_{t^*}) = \theta_a$

(2) $d_{t^*} = 1$

(3) No defender mass is left

IMOA (Increase Defender Mass on Active)

Simultaneously for all $t \in A$, increase d_t, and decrease θ_a, at relative rates that maintain $u_a(t, d_t) = \theta_a$ for all t in A, until one of the following:

(1) For some $t \in U$, $u_a^u(t) = \theta_a$

(2) For some $t \in A$, $d_t = 1$

(3) No defender mass is left

MMNA (Move Attacker Mass to NA)

Choose $t^* \in \arg\max_{t \in NA} v_d(t, a_t)$

Simultaneously move attacker mass from all $t \in A$ to t^*, and decrease θ_d, at relative rates that maintain $v_d(t, a_t) = \theta_d$ for all $t \in A$, until one of the following:

(1) For some $t \in P$, $v_d(t, a_t) = \theta_d$

(2) $v_d(t^*, a_{t^*}) = \theta_d$

(3) $a_{t^*} = 1$

MMDS (Move Attacker Mass to DS)

Choose $t^* \in DS$

Simultaneously move attacker mass from all $t \in A$ to t^*, and decrease θ_d, at relative rates that maintain $v_d(t, a_t) = \theta_d$ for all $t \in A$, until one of the following:

(1) For some $t \in P$, $v_d(t, a_t) = \theta_d$

(2) For some (in fact, all) $t \in A$, $a_t = 0$

(3) $a_{t^*} = 1$

DDT (Decrease the Defender's Threshold)

Decrease θ_d until $\exists t \in NA \cup P$ s.t. $v_d(t, a_t) = \theta_d$

FIGURE 5.2: The algorithm for computing a Nash equilibrium.

ity 1. *We stop when the defender probability on* t_2 *becomes* 1 *and target* t_2 *transitions to the Defender Saturated (DS) phase (Fig. 5.1(h)).*

Move Attacker Mass to Defender Saturated (MMDS) phase: We move attacker mass from t_1 *and* t_3 *to* t_2*. This does not violate the defender's best-response condition because* t_2 *is already fully defended. We stop when the defender becomes indifferent between* $t_1, t_3,$ *and* t_4 *(Fig. 5.1(i)).*

IMOP phase: We increase the defender mass on t_4*, until the attacker becomes indifferent between all targets (Fig. 5.1(j)).*

MMDS phase: We move attacker mass from Active targets t_1, t_3, t_4 *to DS target* t_2 *in the proportions that keep the defender indifferent between the three Active targets, until* t_2 *becomes attacked with probability* 1 *(Fig. 5.1(k)).*

IMOA phase: We add defender mass to t_1, t_3, t_4*, until all defender mass is allocated. After the end of this phase, the algorithm terminates, and the resulting equilibrium profile is an equilibrium of the game with* 3 *defender resources and* 2 *attacker resources (Fig. 5.1(l)).*

5.2.2 Algorithm, Correctness, Runtime

We present the pseudocode for the algorithm in Figure 5.2. The pseudocode contains the exact definitions of the target states. For proof convenience, we will split the Pending target state into two states: $P_1 = \{t \in P : \upsilon_d(t, s_a(t)) < \theta_d\}$, $P_2 = \{t \in P : \upsilon_d(t, s_a(t)) = \theta_d\}$.

Theorem 10. *Throughout the algorithm, the following holds.*

- *UpdateTargetStates always assigns each target to exactly one of the states* $U, A, P_1, P_2, NA, DS, D$.
- *At the end of each phase, if the algorithm does not terminate, then at least one target changes its state.*
- *Each phase terminates.*

Proof sketch. It follows from the state definitions that no target can be in two states at the same time. To prove the theorem, we will first show that each target is assigned a state after the initialization phase, and then we will show which targets change states at the end of each phase.

At the beginning of the algorithm, each target is assigned to exactly one state. The n_a targets for which $s_a(t) = 1$ are assigned to the P state, except that targets with $s_a(t) = 1, u_a^u(t) = \theta_a$, if any, are assigned to the A state. The other $|T| - n_a$ targets are assigned to the U state, except that targets with $s_a(t) = 0, u_a^u(t) = \theta_a$, if any, are assigned to the NA state.

Next, we specify all target state changes for each phase and for each termination criterion within each phase.

Next, for each phase, we list, for every termination criterion of that phase, which targets change states (always a nonempty set, unless the algorithm terminates). It is straightforward to check that one of these criteria will always apply.

IMOP phase: (1) t^* becomes A. (2) t^* becomes D. (3) The algorithm terminates.

IMOA: (1) Every $t \in U$ s.t. $u_a^u(t) = \theta_a$ becomes A. (2) Every $t \in A$ s.t. $s_d(t) = 1$ becomes either DS (if $s_a(t) < 1$) or D (if $s_a(t) = 1$). (3) The algorithm terminates.

MMNA: (1) Every $t \in P$ s.t. $\upsilon_d(t, s_a(t)) = \theta_d$ transitions from P_1 to P_2. (2) Every $t \in NA$ s.t. $\upsilon_d(t, s_a(t)) = \theta_d$ becomes A. (3) t^* transitions from NA to P_1.

MMDS: (1) Every $t \in P$ s.t. $\upsilon_d(t, s_a(t)) = \theta_d$ transitions from P_1 to P_2. (2) If this condition happens, then $\theta_d = 0$, thus $s_a(t) = 0$ for all $t \in P$. Also, $s_a(t) = 0$ for all $t \in A$, and NA is empty. The algorithm will terminate after this phase because $s_a(t) > 0$ implies $s_d(t) = 1$. (3) t^* transitions to D.

DDT: Every $t \in NA$ s.t. $\upsilon_d(t, s_a(t)) = \theta_d$ transitions to A. Every $t \in P$ s.t. $\upsilon_d(t, s_a(t)) = \theta_d$ transitions from P_1 to P_2.

To complete the proof, we also need to show that no target can change state before the phase is over. This can be done in a straightforward way by carefully checking all state definitions. □

Theorem 11. *Throughout the algorithm, the current strategies $\langle s_d, s_a \rangle$ constitute a Nash equilibrium for the current number of defender resources.*

Proof. This follows immediately from the following facts: (1) each target is always in one of the states (Theorem 10), and (2) each state definition implies that the equilibrium condition with respect to the thresholds (beginning of Section 5.2) is satisfied for such a state. □

Theorem 12. *The algorithm terminates after at most $6|T|$ phases, and each phase requires $O(|T|)$ time.*

Proof. We can order the 7 possible states as follows: $U < NA < P_1 < P_2 < A < DS < D$. As we can see from the proof of Theorem 10, after each phase (except the last one), some target changes its state to a later state. Thus the algorithm

terminates after at most $6|T|$ phases. In each phase, we can calculate directly at what point the phase will terminate, though this in general requires examining all $|T|$ targets. □

5.2.3 Interchangeability

While we have shown how to compute a Nash equilibrium efficiently, a defender may still be unconvinced about whether she actually wants to play her corresponding strategy. For example, if she has a commitment advantage where the attacker observes her distribution before acting, she would prefer to play a Stackelberg strategy; we will return to this in Section 5.3. However, even if the attacker cannot observe her distribution, she may worry that she is playing her strategy from the "wrong" equilibrium: in general games, if one player plays according to one equilibrium and the other according to another, the result may be disastrous for both (see the game of chicken). In this section, we alleviate this latter concern, by showing that the security games in this chapter satisfy the *interchange* property: any combination of equilibrium strategies is, in fact, itself an equilibrium. (This was previously shown for a large class of security games with a single attacker resource (Korzhyk et al., 2011).)

Suppose $\sigma = \langle s_d, s_a \rangle$ and $\sigma' = \langle s'_d, s'_a \rangle$ are two NE profiles in a security game with multiple attacker resources. We need to show that $\langle s'_d, s_a \rangle$ and $\langle s_d, s'_a \rangle$ are also NE profiles of the same game. We first prove that for any target, either the defender probability on that target is the same in all equilibrium profiles, or the attacker probability is the same in all equilibrium profiles, or both.

Lemma 13. *If $\sigma = \langle s_d, s_a \rangle$ and $\sigma' = \langle s'_d, s'_a \rangle$ are two NE profiles, then there is no target t for which both (1) the defender probabilities are different in the two profiles and (2) the attacker probabilities are different in the two profiles. In other words, for any target t, at least one of equalities $s_d(t) = s_d(t)'$, $s_a(t) = s'_a(t)$ must hold.*

Proof. The proof is by contradiction. Suppose that there is a target t for which $s_d(t) \neq s_d(t)'$ and $s_a(t) \neq s'_a(t)$. We show that the following four cases must all hold even though two are contradictory.

Case "−−": There exists a target t_1 such that $s_d(t_1)' < s_d(t_1)$ and $s'_a(t_1) < s_a(t_1)$. In profile σ, the attacker's utility $u_a(t_1, s_d(t_1))$ must be greater than or equal to the threshold value θ_a, because $s_a(t_1) > 0$. Similarly, in profile σ', the attacker's utility $u_a(t_1, s_d(t_1)')$ must be less than or equal to the threshold θ'_a, because $s'_a(t_1) < 1$. At the same time, the attacker's utility for attacking t_1 is higher in profile σ' than in profile σ, because $s_d(t_1)' < s_d(t_1)$. Thus, the following three inequalities must hold.

$$\theta_a \leqslant u_a(t_1, s_d(t_1))$$

$$u_a(t_1, s_d(t_1)) < u_a(t_1, s_d(t_1)')$$

$$u_a(t_1, s_d(t_1)') \leqslant \theta'_a$$

It follows from these three inequalities that $\theta'_a > \theta_a$. Because $\sum_t s_a(t) = \sum_t s'_a(t)$ and $s'_a(t_1) < s_a(t_1)$, there must exist a target t_2 such that $s'_a(t_2) > s_a(t_2)$. Since $s'_a(t_2) > 0$, it must be the case that $u_a(t_2, s'_d(t_2)) \geqslant \theta'_a$. Similarly, because $s_a(t_2) < 1$, it must be the case that $u_a(t_2, d_{t_2}) \leqslant \theta_a$. Using the last two inequalities and the fact that $\theta'_a > \theta_a$, it follows that $u_a(t_2, d_{t_2}) < u_a(t_2, s'_d(t_2))$, which implies $s'_d(t_2) < d_{t_2}$. By considering the target t_2, it follows that the case "−+" must also hold.

Case "−+": There is a target t_1 such that $s'_d(t_1) < s_d(t_1)$ and $s'_a(t_1) > s_a(t_1)$. The defender's marginal utility for defending target t_1 must be at or above the threshold θ_d in profile σ (because $s_d(t_1) > 0$) and at or below the threshold θ'_d in profile σ' (because $s'_d(t_1) < 1$). At the same time, since t_1 is attacked with a higher probability in σ' than in σ, it must be that $\upsilon_d(t_1, s'_a(t_1)) > \upsilon_d(t_1, s_a(t_1))$. Thus we have $\theta'_d \geqslant \upsilon_d(t_1, s'_a(t_1)) > \upsilon_d(t_1, s_a(t_1)) \geqslant \theta_d$. Because $\sum_t s_d(t) = \sum_t s_d(t)' = n_d$ and $s'_d(t_1) < s_d(t_1)$, there must be a target t_2 such that $s'_d(t_2) > d_{t_2}$. The defender's marginal utility for defending t_2 must be at or above the threshold θ'_d in profile σ'

(because $s'_d(t_2) > 0$) and at or below the threshold θ_d in profile σ (because $d_{t_2} < 1$). Since $\theta'_d > \theta_d$, it follows that $\upsilon(t_2, s'_a(t_2)) \geqslant \theta'_d > \theta_d \geqslant \upsilon(t_2, s_a(t_2))$, which implies $s'_a(t_2) > s_a(t_2)$. By considering the target t_2, it follows that the case "++" must also hold.

We can also prove the following two implications similarly to the two cases described above, by reversing the roles of equilibria σ and σ':

Case "++": There is a target t_1 such that $s'_d(t_1) > s_d(t_1)$ and $s'_a(t_1) > s_a(t_1)$. If this case holds, then the case "+−" must also hold. This can be proven similarly to the implication "−−" ⇒ "−+", by reversing the roles of equilibria σ and σ'.

Case "+−": There is a target t_1 such that $s'_d(t_1) > s_d(t_1)$ and $s'_a(t_1) < s_a(t_1)$. If this case holds, then the case "−−" must also hold. This can be proven similarly to the implication "−+" ⇒ "++", by reversing the roles of equilibria σ and σ'.

It follows that if at least one of the cases "−−", "−+", "++", "+−" holds, then all of them must hold. But if both "−−" and "++" hold, then both inequalities $\theta'_a > \theta_a$ and $\theta_a > \theta'_a$ must hold, which is impossible. Hence, none of the four cases can hold. □

We now show that in an equilibrium the defender obtains the same marginal utility from all targets that have different defender probabilities in a different equilibrium.

Lemma 14. *Suppose that σ and σ' are two NE profiles, and t_1, t_2 are two targets such that $s_d(t_1) \neq s'_d(t_1)$ and $d_{t_2} \neq s'_d(t_2)$. Then $\upsilon_d(t_1, s_a(t_1)) = \upsilon_d(t_2, s_a(t_2)) = \upsilon_d(t_1, s'_a(t_1)) = \upsilon_d(t_2, s'_a(t_2))$.*

Proof. Because $\sum_t s_d(t) = \sum_t s_d(t)' = n_d$, it is enough to show that $\upsilon_d(t_1, s_a(t_1)) = \upsilon_d(t_2, s_a(t_2))$ holds for any pair of targets t_1, t_2 such that $s_d(t_1) < s'_d(t_1)$ and $d_{t_2} > s'_d(t_2)$. (This is because if (say) $s_d(t_1) < s'_d(t_1)$ and $d_{t_2} < s'_d(t_2)$, there must ex-

ist a third target t_3 with $d_{t_3} > d'_{t_3}$, so that we can then conclude $v_d(t_1, s_a(t_1)) = v_d(t_3, a_{t_3}) = v_d(t_2, s_a(t_2))$.)

In profile σ, the defender can shift her probability from t_2 to t_1, because $d_{t_2} > 0$ and $s_d(t_1) < 1$. Since σ is an equilibrium profile, the defender must not benefit from such a shift of probability. Thus $v_d(t_1, s_a(t_1)) \leqslant v_d(t_2, s_a(t_2))$. Using a similar argument for profile σ', we get $v_d(t_1, s'_a(t_1)) \geqslant v_d(t_2, s'_a(t_2))$. It also follows from Lemma 13 that $s_a(t_1) = s'_a(t_1)$ and $s_a(t_2) = s'_a(t_2)$. Hence, we have $v_d(t_1, s_a(t_1)) \leqslant v_d(t_2, s_a(t_2)) = v_d(t_2, s'_a(t_2)) \leqslant v_d(t_1, s'_a(t_1)) = v_d(t_1, s_a(t_1))$, so it follows that these four quantities are all the same. □

In the following lemma, we show that any defender's NE strategy is a best-response to any attacker's NE strategy.

Lemma 15. *If $\sigma = \langle s_d, s_a \rangle$ and $\sigma' = \langle s'_d, s'_a \rangle$ are two NE profiles, then s'_d is a best-response to s_a.*

Proof. We will show that the defender's utility for playing strategy s'_d against s_a is the same as the defender's utility for playing s_d against s_a. First, note that $u_d(s'_d, s_a) - u_d(s_d, s_a) = \sum_{t:s_d(t)' \neq s_d(t)} [s_d(t)' - s_d(t)]\, v_d(t, s_a(t))$. Consider any target t^* such that $d_{t^*} \neq d'_{t^*}$. Using Lemma 14, we can rewrite the difference in the utilities as follows: $u_d(s'_d, s_a) - u_d(s_d, s_a) = v_d(t_1, s_a(t_1)) \sum_{t:s'_d(t) \neq s_d(t)} [s'_d(t) - s_d(t)] = 0$. The last summation is equal to zero because $\sum_t s_d(t) = \sum_t s'_d(t)$. □

The following lemma can be proven similarly to Lemma 15, by switching from the defender's to the attacker's perspective.

Lemma 16. *If $\sigma = \langle s_d, s_a \rangle$ and $\sigma' = \langle s'_d, s'_a \rangle$ are two NE profiles, then s'_a is a best-response to s_d.*

The interchange property follows from Lemmas 15 and 16.

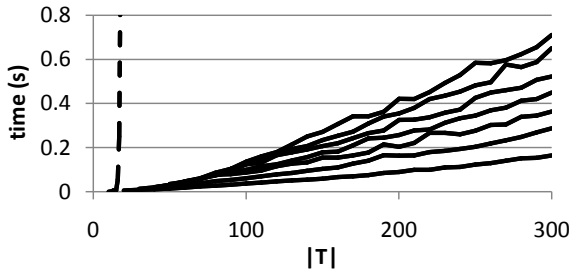

FIGURE 5.3: Solid lines: time to compute NE as a function of $|T|$, for $n_a = n_d = 10, \ldots, 70$. Dashed line: time to compute the normal-form of the game with $n_a = n_d = 10$.

Theorem 17. *If $\sigma = \langle s_d, s_a \rangle$ and $\sigma' = \langle s'_d, s'_a \rangle$ are two NE profiles in a security game with multiple attacker resources, then $\langle s'_d, s_a \rangle$ and $\langle s_d, s'_a \rangle$ are also NE profiles in that game.*

5.2.4 Experimental Results

We now show experimental results for our implementation of the algorithm (Figure 5.3). For given $|T|, n_d, n_a$, we randomly draw u_a^u and u_d^c from $\{1, \ldots, 100\}$, and then we randomly draw u_a^c from $\{0, \ldots, u_a^u - 1\}$ and u_d^u from $\{0, \ldots, u_d^c - 1\}$. Each data point averages over 20 games. As a sanity check, our implementation verified that the strategies computed for each game did constitute a Nash equilibrium. The quadratic runtime of the algorithm is reflected in the experimental results. We note that the numbers of pure strategies for the players are $\binom{|T|}{n_d}$ and $\binom{|T|}{n_a}$, so *any* alternative algorithm that is based on writing out the normal form is doomed to exponential space (and hence, time) requirements. The time to compute the $\binom{|T|}{n_d} \times \binom{|T|}{n_a}$ utility matrix of the normal-form game for $n_d = n_a = 10$ is shown in Figure 5.3 with a dashed line. We can see that our algorithm scales well in the number of targets and attacker resources.

5.3 NP-Hardness of Computing Stackelberg Strategies

We now turn to the problem of computing a defender Stackelberg strategy. Consider the following theorem.

Theorem 18. *In security games with multiple attacker resources, finding an optimal defender Stackelberg strategy is weakly NP-hard. This holds even when the defender has only one resource, and the defender's utility for a target does not depend on whether she has a resource there (that is, $u_d^c(t) = u_d^u(t)$ for all t).*[1]

Proof sketch. We reduce an arbitrary knapsack problem instance—given by k items, where each item j is defined by a pair (w_j, v_j), and we are asked if there is a subset S of the items with $\sum_{j \in S} w_j \leqslant 1$ and $\sum_{j \in S} v_j \geqslant V$—to the following game. We construct a game with $2k$ targets, in which the defender has one resource and the attacker has k resources. Targets $t_1, \ldots, t_k$ correspond to the items in the knapsack, and the utilities are set up as follows for $1 \leqslant i \leqslant k$.

$$u_a^u(t_i) = w_i$$

$$u_a^c(t_i) = w_i - 1$$

$$u_d^c(t_i) = u_d^u(t_i) = -v_i$$

Targets $t_{k+1}, \ldots, t_{2k}$ are "dummy" targets, so that for $k+1 \leqslant i \leqslant 2k$: $u_a^u(t_i) = u_a^c(t_i) = u_d^c(t_i) = u_d^u(t_i) = 0$.

Let the vector s_d represent the defender's strategy, so that $s_d(t_i)$ is the probability of target t_i being covered. If $s_d(t_i) \geqslant w_i$ for $1 \leqslant i \leqslant k$, the attacker attacks a dummy target instead of t_i, thus increasing the defender's utility by v_i.

There exists an attacker pure-strategy best-response to the defender's Stackelberg strategy (with ties broken in the defender's favor) such that the optimal subset S of

[1] We have also found a pseudopolynomial-time algorithm (not presented here) for the special case where $u_d^c(t) = u_d^u(t)$ for all t. Note that this violates the assumption that $u_d^c(t) > u_d^u(t)$. It is easy to modify the utilities by ϵ so that this property holds again and the reduction still works.

the items in the knapsack corresponds to the targets t_i with $1 \leqslant i \leqslant k$ and $a_{t_i} = 0$. It can be shown that the defender can get a utility of at least $V - \sum_{i=1}^{k} v_i$ if and only if the knapsack instance has a solution (with value at least V). $\square$

6

Directions for Future Research

There are a number of directions for future research on the topics discussed in Chapters 2-5.

Regarding the computation of an optimal correlated strategy to commit to, one direction is to try to extend the methodology to game representations other than the normal form. Significant results on the efficient computation of correlated equilibria in succinctly represented games have been obtained, though optimizing over the space of correlated equilibria (which is close to what we do in this thesis) poses more challenges (Papadimitriou and Roughgarden, 2008; Jiang and Leyton-Brown, 2013).

In security games, an important direction for future research is to address the cases in which computing a Stackelberg strategy is NP-hard. Can we find algorithms that, although they require exponential time in the worst case, solve typical instances fast? Can we identify additional restrictions on the game so that the problem becomes polynomial-time solvable (Letchford and Conitzer, 2013)? Are there good polynomial-time approximation algorithms, or anytime algorithms that find a reasonably good solution fast (Jain et al., 2013)? Another direction for future research is to consider security games with incomplete information (Bayesian games) or mul-

tiple time periods (extensive-form of stochastic games). In unrestricted games, these aspects can lead to additional complexity (Conitzer and Sandholm, 2006; von Stengel and Zamir, 2010; Letchford et al., 2009; Letchford and Conitzer, 2010; Letchford et al., 2012).

As for security games with multiple attacker resources, one natural question is whether the algorithm that we presented in this thesis can be generalized to richer settings. For example, is it possible to compute Nash equilibria efficiently in cases where either defender resources or attacker resources (or both) are heterogeneous? Can we efficiently compute them in (restricted) settings with schedules?

Bibliography

An, B., Shieh, E., Yang, R., and Tambe, M. (2013), "PROTECT - A Deployed Game Theoretic System for Strategic Security Allocation for the United States Coast Guard," *AI Magazine*.

Aumann, R. (1974), "Subjectivity and Correlation in Randomized Strategies," *Journal of Mathematical Economics*, 1, 67–96.

Bhattacharya, S., Conitzer, V., and Munagala, K. (2011), "Approximation algorithm for security games with costly resources," in *Workshop on Internet and Network Economics*, pp. 13–24, Singapore.

Birkhoff, G. (1946), "Tres observaciones sobre el algebra lineal," *Univ. Nac. Tucumn Rev, Ser. A, no. 5*, pp. 147–151.

Chang, C.-S., Chen, W.-J., and Huang, H.-Y. (2001), "Coherent Cooperation Among Communicating Problem Solvers," *IEEE Transactions on Communications*, 49, 1145–1147.

Chen, X., Deng, X., and Teng, S.-H. (2009), "Settling the complexity of computing two-player Nash equilibria," *Journal of the ACM*, 56.

Conitzer, V. and Sandholm, T. (2006), "Computing the Optimal Strategy to Commit to," in *Proceedings of the ACM Conference on Electronic Commerce (EC)*, pp. 82–90, Ann Arbor, MI, USA.

Conitzer, V. and Sandholm, T. (2008), "New Complexity Results about Nash Equilibria," *Games and Economic Behavior*, 63, 621–641.

Dantzig, G. B. and Wolfe, P. (1960), "Decomposition principle for linear programs," *Operations research*, 8, 101–111.

Daskalakis, C., Goldberg, P., and Papadimitriou, C. H. (2009), "The Complexity of Computing a Nash Equilibrium," *SIAM Journal on Computing*, 39, 195–259.

Dulmage, L. and Halperin, I. (1955), "On a theorem of Frobenius-Konig and J. von Neumann's game of hide and seek," *Trans. Roy. Soc. Canada III*, 49, 23–29.

Galil, Z., Micali, S., and Gabow, H. (1986), "An $O(EV \log V)$ Algorithm for Finding a Maximal Weighted Matching in General Graphs," *SIAM J. Comput.*, 15, 120–130.

Gilboa, I. and Zemel, E. (1989), "Nash and correlated equilibria: Some complexity considerations," *Games and Economic Behavior*, 1, 80–93.

Halvorson, E., Conitzer, V., and Parr, R. (2009), "Multi-step Multi-sensor Hider-Seeker Games," in *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 159–166, Pasadena, CA, USA.

Jain, M., Kardes, E., Kiekintveld, C., Ordóñez, F., and Tambe, M. (2010), "Security Games with Arbitrary Schedules: A Branch and Price Approach," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 792–797, Atlanta, GA, USA.

Jain, M., Kiekintveld, C., and Tambe, M. (2011), "Quality-bounded Solutions for Finite Bayesian Stackelberg Games: Scaling up," in *Proceedings of the Tenth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 997–1004, Taipei, Taiwan.

Jain, M., Conitzer, V., and Tambe, M. (2013), "Security Scheduling for Real-world Networks," in *AAMAS*, Saint Paul, Minnesota, USA.

Jiang, A. X. and Leyton-Brown, K. (2013), "Polynomial-time Computation of Exact Correlated Equilibrium in Compact Games," *Games and Economic Behavior*.

Jiang, A. X., Yin, Z., Zhang, C., Tambe, M., and Kraus, S. (2013), "Game-theoretic Randomization for Security Patrolling with Dynamic Execution Uncertainty," in *Proceedings of the Twelfth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Saint Paul, Minnesota, USA.

Kiekintveld, C., Jain, M., Tsai, J., Pita, J., Ordóñez, F., and Tambe, M. (2009), "Computing Optimal Randomized Resource Allocations for Massive Security Games," in *Proceedings of the Eighth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 689–696, Budapest, Hungary.

Korzhyk, D., Conitzer, V., and Parr, R. (2010), "Complexity of Computing Optimal Stackelberg Strategies in Security Resource Allocation Games," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 805–810, Atlanta, GA, USA.

Korzhyk, D., Yin, Z., Kiekintveld, C., Conitzer, V., and Tambe, M. (2011), "Stackelberg vs. Nash in Security Games: An Extended Investigation of Interchangeability, Equivalence, and Uniqueness," *Journal of Artificial Intelligence Research*, 41, 297–327.

Lemke, C. and Howson, J. (1964), "Equilibrium points of bimatrix games," *Journal of the Society of Industrial and Applied Mathematics*, 12, 413–423.

Letchford, J. and Conitzer, V. (2010), "Computing Optimal Strategies to Commit to in Extensive-Form Games," in *Proceedings of the ACM Conference on Electronic Commerce (EC)*, pp. 83–92, Cambridge, MA, USA.

Letchford, J. and Conitzer, V. (2013), "Solving Security Games on Graphs via Marginal Probabilities," in *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, Bellevue, WA, USA.

Letchford, J. and Vorobeychik, Y. (2012), "Computing Optimal Security Strategies for Interdependent Assets," in *Uncertainty in Artificial Intelligence*, Catalina Island, CA.

Letchford, J., Conitzer, V., and Munagala, K. (2009), "Learning and Approximating the Optimal Strategy to Commit to," in *Proceedings of the Second Symposium on Algorithmic Game Theory (SAGT-09)*, pp. 250–262, Paphos, Cyprus.

Letchford, J., MacDermed, L., Conitzer, V., Parr, R., and Isbell, C. (2012), "Computing Optimal Strategies to Commit to in Stochastic Games," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 1380–1386, Toronto, ON, Canada.

McKelvey, R. D., McLennan, A. M., and Turocy, T. L. (2004), "Gambit: Software Tools for Game Theory, Version 0.97.1.5," .

McMahan, H. B., Gordon, G. J., and Blum, A. (2003), "Planning in the Presence of Cost Functions Controlled by an Adversary," in *International Conference on Machine Learning (ICML)*, pp. 536–543, Washington, DC, USA.

Nash, J. (1950), "Equilibrium points in N-person games," *Proceedings of the National Academy of Sciences*, 36, 48–49.

Nudelman, E., Wortman, J., Leyton-Brown, K., and Shoham, Y. (2004), "Run the GAMUT: A Comprehensive Approach to Evaluating Game-Theoretic Algorithms," in *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 880–887, New York, NY, USA.

Papadimitriou, C. H. and Roughgarden, T. (2008), "Computing Correlated Equilibria in Multi-Player Games," *Journal of the ACM*, 55.

Paruchuri, P., Pearce, J. P., Marecki, J., Tambe, M., Ordóñez, F., and Kraus, S. (2008), "Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games," in *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 895–902, Estoril, Portugal.

Pita, J., Jain, M., Ordóñez, F., Portway, C., Tambe, M., and Western, C. (2009), "Using game theory for Los Angeles airport security," *AI Magazine*, 30, 43–57.

Sandholm, T. (2010), "The State of Solving Large Incomplete-Information Games, and Application to Poker," *AI Magazine*, 31, 13–32, Special Issue on Algorithmic Game Theory.

Sandholm, T., Gilpin, A., and Conitzer, V. (2005), "Mixed-Integer Programming Methods for Finding Nash Equilibria," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 495–501, Pittsburgh, PA, USA.

Tsai, J., Rathi, S., Kiekintveld, C., Ordonez, F., and Tambe, M. (2009), "IRIS - A Tool for Strategic Security Allocation in Transportation Networks," in *Proceedings of the Eighth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 37–44, Budapest, Hungary.

von Neumann, J. (1928), "Zur Theorie der Gesellschaftsspiele," *Mathematische Annalen*, 100, 295–320.

von Stackelberg, H. (1934), *Marktform und Gleichgewicht*, pp. 58–70, Springer, Vienna.

von Stengel, B. and Zamir, S. (2010), "Leadership Games with Convex Strategy Sets," *Games and Economic Behavior*, 69, 446–457.

Biography

Dmytro Korzhyk was born on February 3, 1986 in Severodonetsk, Ukraine. He grew up in Vinnytsia, Ukraine. He obtained his Bachelors degree in 2006 and his Masters degree in 2007, both in Computer Science from Vinnytsia National Technical University. He enrolled in the Ph.D. program in Computer Science at Duke University in 2008.