

# Computational Aspects of Stackelberg Games

by

Joshua Letchford

Department of Computer Science  
Duke University

Date: \_\_\_\_\_

Approved:

\_\_\_\_\_  
Vincent Conitzer, Supervisor

\_\_\_\_\_  
Kamesh Munagala

\_\_\_\_\_  
Ronald Parr

\_\_\_\_\_  
Atila Abdulkadiroglu

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

Computational Aspects of Stackelberg Games

by

Joshua Letchford

Department of Computer Science  
Duke University

Date: \_\_\_\_\_

Approved:

---

Vincent Conitzer, Supervisor

---

Kamesh Munagala

---

Ronald Parr

---

Atila Abdulkadiroglu

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 Joshua Letchford  
All rights reserved

# Abstract

Game theory involves the study of how self-interested agents interact in various settings. Perhaps the best-known game theoretic solution concept is that of Nash equilibrium. However, Nash equilibrium makes a number of assumptions, such as rationality of all of the players and the ability of the players to choose the same equilibrium when more than one exists. Because of these assumptions, it is unclear if simply solving for Nash equilibrium is always the correct thing to do. In some settings, one player can instead credibly commit to a strategy, and communicate this to the other player, before the other player can make a decision. This has become known as a Stackelberg game. Computing optimal strategies to commit to in normal-form or Bayesian Stackelberg games is a topic that has recently been gaining attention, in part due to the application of such algorithms in various security and law enforcement scenarios. My work on Stackelberg games falls into three main areas.

First, I focus on general games, where we give efficient algorithms and hardness results for Bayesian, extensive-form and stochastic games. In each of these settings we study the relationship between different modeling assumptions and the tractability of finding an optimal strategy to commit to. For Bayesian games our results are mainly negative; we show that not only are the problems here NP-hard, but in many cases they are also inapproximable. Our results for extensive-form games are more mixed; we are able to give polynomial time algorithms for four cases. However, we also show that if we relax the assumptions made in these four cases, then the problem usually becomes NP-hard. Finally, our results for stochastic games are again somewhat negative, as we show that

the problem is NP-hard in most reasonable cases. However, here we are also able to give an approximation algorithm to compute optimal commitment strategies in a setting where correlation is allowed.

I next focus on Stackelberg security games. Stackelberg security games usually involve the scheduling of scarce defense resources to cover some subset of potential targets. We first study methods for going from marginal solutions (which ignore overlapping coverage between different schedules) to valid mixed commitment strategies in graphical settings. Here we are able to characterize two new classes of games where mixed strategies corresponding to the marginal probabilities are guaranteed to exist, and give algorithms for finding them. Next, we offer a simple model of interdependencies between nodes in a network based on probabilistic failure cascades, extending the well-known independent cascade model of the spread of infectious diseases or ideas. We give an algorithm for this problem and experimentally show that this algorithm scales to realistic security settings and outperforms the state-of-the-art alternatives. Finally, we create an approach for optimal interdiction of attack plans. We show how to model an attack plan interdiction problem as a large-scale integer linear program similar to an integer programming approach for solving partial satisfaction planning problems. We also give several oracle-based approaches for solving this and then evaluate them experimentally.

Third, I analyze *how much* benefit a player can derive from commitment in various types of games, in a quantitative sense that is similar to known concepts such as the value of mediation and the price of anarchy. To do this we introduce and study the *value of pure commitment* (the benefit of committing to a pure strategy), the *value of mixed commitment* (the benefit of committing to a mixed strategy), and the *mixed vs. pure commitment* ratio (how much can be gained by committing to a mixed strategy rather than a pure one).

# Contents

<b>Abstract</b>	<b>iv</b>
<b>List of Figures</b>	<b>x</b>
<b>Acknowledgements</b>	<b>xiv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background on game theory . . . . .	2
1.2 Stackelberg games . . . . .	5
1.3 Related work . . . . .	8
1.4 Contributions . . . . .	10
<b>2 Commitment in Bayesian Games</b>	<b>14</b>
2.1 Computing optimal strategies with finitely many types . . . . .	16
2.2 Computing worst-case optimal strategies with finitely many types . . . . .	19
2.3 Optimizing for the worst type with ranges . . . . .	22
2.4 Worst-case optimization for linked range types . . . . .	24
2.5 Mixed integer program formulations of BOFT and WOFT . . . . .	26
<b>3 Commitment in Extensive-Form Games</b>	<b>28</b>
3.1 Positive results . . . . .	32
3.1.1 Case 1. No chance, pure strategies, trees, no costs or restrictions, perfect information . . . . .	33
3.1.2 Case 2. No chance, pure strategies, trees, perfect information . . . . .	34

3.1.3	Case 3. No chance, pure strategies, two players, no costs or restrictions, perfect information . . . . .	37
3.1.4	Case 4. No chance, trees, two-player, no costs or restrictions, perfect information . . . . .	38
3.2	Negative results . . . . .	42
3.2.1	Case 1. Chance nodes . . . . .	43
3.2.2	Case 2. Three-players, mixed strategy commitment . . . . .	47
3.2.3	Case 3. Two players, imperfect information, pure or mixed strategy commitment . . . . .	51
3.2.4	Case 4. Two-players, DAG, mixed strategy commitment . . . . .	55
3.2.5	Case 5. Three-players, DAG, pure or mixed strategies . . . . .	59
3.2.6	Case 6. Two-players, DAG, restrictions, pure or mixed strategies . . . . .	61
<b>4</b>	<b>Commitment in Stochastic Games</b>	<b>65</b>
4.1	Stochastic games . . . . .	66
4.2	Commitment and signals . . . . .	67
4.3	Value of correlation and commitment . . . . .	69
4.4	Hardness results . . . . .	71
4.5	Computing commitment equilibria . . . . .	75
<b>5</b>	<b>Solving Security Games on Graphs via Marginal Probabilities</b>	<b>78</b>
5.1	Background . . . . .	79
5.2	Dimensions of the problem . . . . .	82
5.3	Positive results . . . . .	83
5.3.1	Heterogeneous, set of trees, paths from the root . . . . .	84
5.3.2	Homogeneous, path graph, general paths . . . . .	85
5.4	Negative results . . . . .	87
5.4.1	Heterogeneous resources, a path graph, schedules are edges . . . . .	87
5.4.2	Homogeneous, general graph, paths from the root . . . . .	90

5.4.3	Tree graph, paths that pass through the root . . . . .	90
<b>6</b>	<b>Computing Optimal Security Strategies for Interdependent Assets</b>	<b>93</b>
6.1	Computing optimal randomized security configurations . . . . .	95
6.2	Incorporating network structure . . . . .	96
6.2.1	A General Model of Interdependencies . . . . .	96
6.2.2	Cascading Failures Model . . . . .	98
6.2.3	Computing Expected Utilities . . . . .	99
6.2.4	The Significance of Capturing Interdependence . . . . .	103
6.2.5	Incorporating Uncertainty about the Network . . . . .	103
6.3	Experiments . . . . .	104
6.3.1	Sampling Efficiency . . . . .	105
6.3.2	Scalability . . . . .	105
6.3.3	Comparison to State-of-the-Art Alternatives . . . . .	105
6.4	Applications to interdependent security analysis . . . . .	107
6.5	The impact of uncertainty . . . . .	108
6.5.1	The Impact of Marginal Defense Cost . . . . .	108
6.5.2	Resilience to Targeted Attacks: Impact of Network Structure . . .	109
<b>7</b>	<b>Optimal Interdiction of Attack Plans</b>	<b>113</b>
7.1	Integer programming for classical planning . . . . .	114
7.2	Deterministic plan interdiction . . . . .	118
7.2.1	The Plan Interdiction Problem . . . . .	118
7.2.2	Integer Programming Formulation for Optimal Plan Interdiction .	121
7.2.3	Scaling Up with Constraint Generation . . . . .	122
7.2.4	Optimistic Constraint Generation . . . . .	124
7.2.5	Tie breaking . . . . .	125

7.2.6	Heuristic Plan Interdiction . . . . .	126
7.3	Experiments with deterministic plan interdiction . . . . .	128
7.4	Dealing with uncertainty . . . . .	131
7.4.1	Uncertainty about Attacker’s Capabilities, Costs, and Goals . . . . .	131
7.4.2	Uncertain Attack Execution with Retry . . . . .	132
7.4.3	MDP Interdiction Problem . . . . .	132
<b>8</b>	<b>Value of Commitment</b>	<b>135</b>
8.1	A formal definition of value . . . . .	137
8.2	Normal-form and symmetric normal-form games . . . . .	139
8.3	Extensive-form games . . . . .	141
8.4	Security games . . . . .	143
8.5	Routing games . . . . .	144
8.6	Quadratic atomic routing games . . . . .	150
8.7	Arbitrary cost atomic routing . . . . .	155
<b>9</b>	<b>Conclusion</b>	<b>158</b>
	<b>Bibliography</b>	<b>162</b>
	<b>Biography</b>	<b>168</b>

# List of Figures

1.1	The game Rock-Paper-Scissors in normal form. . . . .	2
1.2	A general-sum game with more than one equilibrium. . . . .	4
1.3	An example normal-form game . . . . .	5
1.4	The extensive-form representation of the game in figure 1.3 with commitment. . . . .	6
3.1	Example of DAG depiction methods. . . . .	30
3.2	Summary of extensive form results. At any node, some of the aspects of the problem are fixed, others are not. Every leaf node states whether the problem is in P or NP-hard at that leaf; this result then applies for <i>any</i> way of fixing the remaining aspects of the problem. . . . .	32
3.3	Two-player game with chance nodes used in the hardness reduction of Theorem 10. . . . .	43
3.4	Pseudo-chance node gadget used in Theorem 11 and Theorem 13. . . . .	45
3.5	Three-player tree with pseudo-chance node used in the hardness reduction of Theorem 11. . . . .	48
3.6	Two-player game of imperfect information used in the hardness reduction of Theorem 12. The notation (X) next to a node indicates that the node is part of the information set X. In $t^{C_j}$ , $+x_i$ appears in $C_j$ . . . . .	52
3.7	Two-player DAG game used in the hardness reduction of Theorem 13. Note that $l^1(C_j)$ is the first literal in clause $C_j$ , $l^{\text{last}}(C_j)$ is the last literal in clause $C_j$ , and $x(l)$ is the variable corresponding to literal $l$ . . . . .	56
3.8	Three-player DAG used in the hardness reduction of Theorem 14. Note that $t_{\text{last}}^{x_{C_j^+}}$ is the variable subtree corresponding to the last positive literal appearing in clause $C_j$ . . . . .	60

3.9	Two-player DAG with restrictions used in the hardness reduction of Theorem 15. Note that $t^{x_{\text{last}}^{C_j^+}}$ is the variable subtree corresponding to the last positive literal appearing in clause $C_j$ , etc. . . . .	62
4.1	Example stochastic game. . . . .	66
4.2	Example game where signaling must occur early, but not too early. . . . .	68
4.3	Game where commitment offers an unbounded advantage over correlation. . . . .	69
4.4	Game where correlation offers an unbounded advantage over commitment. . . . .	70
4.5	Overview of hardness results. $h$ represents the amount of history that player 1 can remember. . . . .	71
4.6	Stochastic game used in the hardness reduction of Theorem 18. . . . .	72
4.7	Stochastic game used in the hardness reduction of Theorem 20. . . . .	75
5.1	Summary of results. Every entry states, before the /, whether the Birkhoff-von Neumann property holds (“Yes” or “No”), that is, whether marginal probabilities can always be realized. (“Yes*” indicates that we only prove this for the marginal probabilities that result from our direct algorithm.) After the /, it states the complexity of finding an optimal Stackelberg strategy. All of the positive results hold even if the graph consists of multiple lines/trees/DAGs (that each have their own root in the rooted case), and all the negative results hold even if there is only one component. . . . .	82
5.2	. . . . .	88
6.1	Left: Comparison between our approach (“with graph info”) and one assuming independence (“without graph info”) using the ER(0.1) generative model. Middle: Comparison to the degree-based heuristic on PA graphs. Right: Comparison to the degree-based heuristic on the AS graph. . . . .	106
6.2	Left: Comparison between assuming only two configurations, and allowing the defender to consider three alternatives. Right: Comparison between a solution which assumes that failures are only due to attacks, and an optimal solution, when failures are actually random. Comparisons use PA graphs. . . . .	107
6.3	The impact of noise on PA networks. Left: when $p_{t,t'} = 0.5$ ; Right: when $p_{t,t'} = 0.1$ . . . . .	109

6.4	Expected loss, cost, and their sum in (a) 100-node ER(0.2), (b) 100-node PA, (c) 18-node critical infrastructure, and (d) 66-node core of the Fedwire networks as defense cost increases. The results for ER and PA are averages over 100 stochastic realizations of these networks. . . . .	110
6.5	Left: Expected total loss: comparison across different network structures. Right: Expected defender disutility in the generalized PA model as we vary $\mu$ (keeping average degree fixed at 2). ER is also shown for comparison.	111
6.6	Expected loss, cost, and their sum in 100-node Erdos-Renyi networks as a function of network density (equivalently, expected degree). . . . .	112
7.1	Example attack graph. Boxes correspond to initial attacker capabilities, ovals are attack actions, and diamonds are attacker goals. An optimal attack plan is highlighted in red. . . . .	117
7.2	Example interdiction plan: actions that are blocked are colored in blue, and the final attack plan (circumvention) is highlighted in red. . . . .	124
7.3	(a) Runtime comparison for several variants of constraint generation. (b) DPIP runtime vs. optimal planning runtime. (c) Runtime and (d) utility comparison of tie breaking approaches. . . . .	127
7.4	Comparison of the greedy heuristic (Algorithm 2) to optimal in terms of defender utility (left) and runtime (right). . . . .	130
7.5	Runtime (left) and defender utility (right) of Bayesian plan interdiction as a function of the number of attacker types. . . . .	131
7.6	Runtime comparison between <i>MDPIP</i> and <i>DPIP</i> . . . . .	134
8.1	Game for $VoPC^{ISD}(NF_{2 \times 2})$ and $VoPC^{ISD}(NF_{2 \times 2})$ . . . . .	140
8.2	Game for $MvP^{ISD}(NF_{2 \times 2})$ . . . . .	140
8.3	Game for $VoPC^{ISD}(SNF_{3 \times 3})$ and $VoMC^{ISD}(SNF_{3 \times 3})$ . . . . .	141
8.4	Game for $MvP^{ISD}(SNF_{3 \times 3})$ . . . . .	141
8.5	Game for $VoPC^{SPNE}(PIEF)$ and $VoMC^{SPNE}(PIEF)$ . . . . .	142
8.6	Game for $MvP^{SPNE}(PIEF)$ . . . . .	142
8.7	Game for $VoPC^{UniqueNash}(SSG_{2 \times 2})$ and $VoMC^{UniqueNash}(SSG_{2 \times 2})$ . . . . .	144
8.8	Game for $MvP^{UniqueNash}(SSG_{2 \times 2})$ . . . . .	144
8.9	2-player <i>LAR</i> routing game for <i>VoPC</i> and <i>VoMC</i> . . . . .	146

8.10	3-player <i>LAR</i> routing game for <i>VoPC</i> and <i>VoMC</i> . . . . .	146
8.11	4-player <i>LAR</i> game for <i>VoPC</i> and <i>VoMC</i> . . . . .	148
8.12	3-player <i>LAR</i> game for <i>MvP</i> . . . . .	150
8.13	3-player <i>QAR</i> game for <i>VoPC</i> and <i>VoMC</i> . . . . .	152
8.14	3-player <i>QAR</i> game for <i>MvP</i> . . . . .	152
8.15	3-player <i>KAR</i> game for <i>VoPC</i> and <i>VoMC</i> . . . . .	155
8.16	3-player <i>KAR</i> game for <i>MvP</i> . . . . .	155
8.17	2-player $AAR_{2\text{ player}}$ game for <i>VoPC</i> and <i>VoMC</i> . . . . .	156
8.18	2-player $AAR_{2\text{ player}}$ game for <i>MvP</i> . . . . .	156

# Acknowledgements

I would first like to thank my advisor, Prof. Vincent Conitzer for introducing me to the field of computational game theory, for his availability and support throughout the past six years, and for his great advice on research, jobs and many other topics. I was very lucky to have been able to work with someone as talented as Vince who also always has had the time to help when I've needed it.

I would like to thank my other committee members for always being a good source of feedback on both papers and presentations and for asking great questions that have helped guide my research in interesting directions.

I would also like to thank Yevgeniy Vorobeychik at Sandia National Laboratories for his guidance during my internships there. I've benefited greatly from my time there working with him.

Finally, I would like to thank my parents, for always being supportive of me, for making the sacrifices to open opportunities for me, and for always encouraging me to challenge myself.

This dissertation is based on work supported by NSF Awards IIS-0953756, IIS-0812113, and CCF-1101659, ARO MURI Grant W911NF-11-1-0332, ARO W911NF-09-1-0459, ARO W911NF-11-1-0332 and an Alfred P. Sloan fellowship.

# 1

## Introduction

Game theory involves the study of how self-interested agents (*players*) interact in various settings. Traditionally, the most common solution concept studied is Nash equilibrium. A set of strategies is a Nash equilibrium, if no player can unilaterally deviate and expect to increase their expected utility. A strategy can be either pure, meaning a deterministic action by a player, or mixed, which allows for randomization over the deterministic actions of the player. If strategies are allowed to be mixed then it is known that every finite game has at least one Nash equilibrium.

There exist a number of related solution concepts, such as *correlated* equilibrium [5] where the players can observe a signal before making any decision, allowing for coordination that would be impossible otherwise. Another equilibrium concept, *Stackelberg* equilibrium [62], changes the model in a different fashion, by allowing for an asymmetry in the order in which the players make their choices and additionally allowing the player who moves first to communicate how she plans on moving. While it may initially seem unintuitive that tying one's hands before the other players move could be beneficial, it is well known result in game theory that this is indeed so.

Much of the driving force behind the recent research into the computational aspects of

Stackelberg games is a number of applications to real security problems. The first of these was the ARMOR project [22, 48], which has been employed since 2007 for scheduling vehicle checkpoints and canine patrols at Los Angeles International Airport. The next domain considered involved placing federal air marshals on international flights (IRIS [57]) and has been deployed by the Federal Air Marshal Service since 2009. In 2011 PROTECT [55] was deployed to schedule Coast Guard patrols at the port of Boston. Two additional applications still in evaluation are TRUSTS [67] for scheduling patrols on the Los Angeles subway and light rail system to catch ticketless travelers, and GUARDS [50], which is an attempt to schedule airport security on a national level.

This dissertation studies the computational problems involved in finding Stackelberg equilibria over a wide range of game representations. Our main focus is to show which assumptions in each of these games are necessary for there to exist polynomial time algorithms, by finding these algorithms and when possible, showing that when these assumptions are relaxed, these problems become NP-hard.

## 1.1 Background on game theory

Perhaps the most basic representation of a game is the *normal form*. In the normal-form representation, every player's pure strategies are explicitly listed, and for every combination of pure strategies, every player's utility is explicitly listed. An example of the game known as Rock-Paper-Scissors (RPS) in normal form appears in Figure 1.1

	<i>Rock</i>	<i>Paper</i>	<i>Scissors</i>
<i>Rock</i>	(0,0)	(-1,1)	(1,-1)
<i>Paper</i>	(1,-1)	(0,0)	(-1,1)
<i>Scissors</i>	(0,0)	(1,-1)	(0,0)

FIGURE 1.1: The game Rock-Paper-Scissors in normal form.

In this game, and other normal form games, we will refer to the player that chooses

between the row (column) actions as player 1 (2). Each player receives the payoffs denoted for the combination of actions played. For example, when player 1 chooses Rock and player 2 chooses Scissors, then player 1 receives 1 and player 2 receives  $-1$ . Generally, we will let  $A_1$  be the set of player 1 actions in the game, and let  $A_2$  be the set of player 2 actions. Player 1's (2's) utility is given by a function  $u_1 : A_1 \times A_2 \rightarrow \mathbb{R}$  ( $u_2$ ). RPS is what is known as a zero-sum game, where the payoffs for each possible outcome in the game sum to 0. Recall that Nash equilibrium is a solution concept that assumes that each player knows the equilibrium strategy of the other player, and given this information has no incentive to change their own strategy. This game has no pure strategy Nash equilibrium, as the nature of RPS means that any strategy has a corresponding strategy for the other player that guarantees a win. For example, Rock punishes a player that always plays Scissors. A mixed strategy is an assignment of probability over a player's set of pure strategies. Let  $\sigma_1$  denote a mixed strategy for player 1, and  $\sigma(a_1)$  the probability that  $\sigma_1$  places on action  $a_1$ . The unique mixed-strategy Nash equilibrium for this game is when both players equally mix between the three strategies. In zero-sum games, we can find a mixed strategy Nash equilibrium with the Minimax theorem [61]. The Minimax theorem states that for every two-player, zero-sum game with a finite number of strategies:

$$\max_{\sigma_1} \min_{a_2} u_1(\sigma_1, a_2) = \min_{\sigma_2} \max_{a_1} u_1(a_1, \sigma_2)$$

That is, there exists a strategy for player 1 that guarantees a payoff of  $V$  against any response by player 2, and there exists a strategy for player 2 that guarantees a payoff of  $-V$  against any response by player 1 (as it guarantees at most a payoff of  $V$  for player 1). Since the utilities of the two players must add up to zero, the existence of each of these values implies the other value is optimal. This is closely related to the concept of linear programming duality. In fact, computing minimax strategies is equivalent to linear programming [15, 1].

However, not all games are zero-sum. General-sum games often have more than one

	<i>Left</i>	<i>Right</i>
<i>Up</i>	(1,2)	(0,0)
<i>Down</i>	(0,0)	(2,1)

FIGURE 1.2: A general-sum game with more than one equilibrium.

equilibrium, leading to the *equilibrium selection problem*. An example of this appears in Figure 1.2. While both  $\{L,U\}$  and  $\{R,D\}$  are equilibria in this game, it is non-obvious which should be chosen. While player 1 would prefer that  $\{L,U\}$  be chosen, player 2 would instead prefer  $\{R,D\}$ .

Sometimes it is possible to narrow down the set of strategies that can be played in any Nash equilibrium through a process known as iterated dominance. A strategy for player 1 ( $\sigma_1$ ) is said to weakly dominate another strategy ( $\sigma'_1$ ) if:

$$\forall s_2 u_1(\sigma_1, s_2) \geq u_1(\sigma'_1, s_2) \text{ and } \exists u_1(\sigma_1, s_2) > u_1(\sigma'_1, s_2)$$

and is said to strictly dominate if:

$$\forall s_2 u_1(\sigma_1, s_2) > u_1(\sigma'_1, s_2)$$

Any solution that is strictly dominated by another solution cannot be a part of any Nash equilibrium, although the same cannot be said of a strategy that is weakly dominated. Iterated dominance takes advantage of this fact by removing any dominated strategies from the game, then checking if any strategies are dominated in the new game. If this process converges to a single strategy for each player, then this pair of strategies is guaranteed to be a Nash equilibrium. When only strictly dominated strategies are removed in this fashion and the process converges, the Nash equilibrium is guaranteed to be unique.

Unfortunately, this technique fails to converge to a single strategy for each player in many games (such as the game in Figure 1.2). In fact, finding a Nash equilibrium in a two-player general-sum game is PPAD-complete [16, 9] (FIXP-complete for 3+ players [18]) and finding an optimal Nash equilibrium (for just about any reasonable defi-

nition of “optimal”—for instance, maximizing the sum of the players’ utilities) is NP-hard [21, 14]. In fact, it is not even possible to find an equilibrium that is approximately optimal in polynomial time, unless  $P=NP$ , even in two-player games [14].

## 1.2 Stackelberg games

Recall that in a Stackelberg equilibrium, one of the players (the leader) is said to have a commitment advantage over the other player, meaning she are able to announce what action they will be taking in a believable fashion. Consider the following example of a normal-form game to illustrate how commitment can be advantageous to the leader:

**Example 1.** (known) Consider the normal-form game in Figure 1.3.

	<i>L</i>	<i>R</i>
<i>U</i>	(1,1)	(3,0)
<i>D</i>	(0,0)	(2,1)

FIGURE 1.3: An example normal-form game

*For the case where the players move simultaneously (no ability to commit), the unique Nash equilibrium is (U,L): U strictly dominates D, so that the game is solvable by iterated strict dominance. So, player 1 (the row player) receives utility 1. However, now suppose that player 1 has the ability to commit. Then, she is better off committing to play D, which will incentivize player 2 to play R, resulting in a utility of 2 for player 1. The situation gets even better for player 1 if she can commit to a mixed strategy: in this case, she can commit to the mixed strategy  $(.5 - \epsilon, .5 + \epsilon)$ , which still incentivizes player 2 to play R, but now player 1 receives an expected utility of  $2.5 - \epsilon$ . To ensure the existence of optimal strategies, we assume (as is commonly done [13, 47]) that player 2 breaks ties in player 1’s favor, so that the optimal strategy for player 1 to commit to is  $(.5, .5)$ , resulting in a utility*

of 2.5. (Note that there is never a reason for player 2 to randomize, since he effectively faces a single-agent decision problem.)

In the above example, commitment to both *pure* and *mixed* strategies improve the leader's utility when compared to Nash equilibrium. In fact, it can be shown that, in a sense, optimal commitment to a mixed strategy never hurts a player (relative to any Nash equilibrium or even any correlated equilibrium) [64]. (A rough intuition is that a player can simply commit to her equilibrium strategy.) In contrast, committing to a pure strategy is not always beneficial; for example, consider the game of Rock-Paper-Scissors discussed earlier.

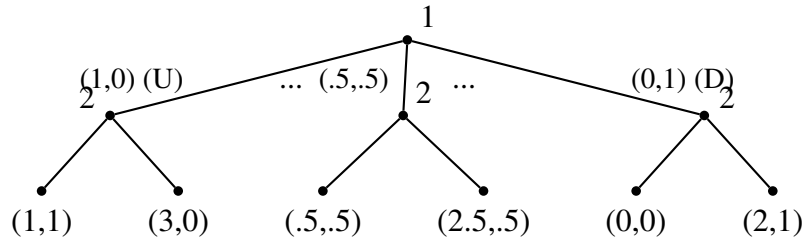


FIGURE 1.4: The extensive-form representation of the game in figure 1.3 with commitment.

In addition, one might argue that the normal-form is not the correct representation for this game. In fact, the time structure of games is often represented by the *extensive form*. The above game can be represented as the extensive-form game in Figure 1.4. While this is a conceptually useful representation, from a computational perspective it is not helpful: player 1 has an infinite number of strategies, hence (the naïve representation of) the tree has infinite size. It should be emphasized that committing to a mixed strategy is *not* the same as randomizing over which pure strategy to commit to; in fact, there is no reason to randomize over which strategy to commit to. Thus, from a computational viewpoint, it makes more sense to operate directly on the normal form.

Next, we review the standard algorithm for computing a Stackelberg mixed strategy in two-player normal-form games [13, 64]. This algorithm creates a separate linear program

(LP) for each follower pure strategy  $a_f^* \in A_f$ . An optimal solution of this LP gives the optimal leader strategy and utility, under the constraint that  $a_f^*$  is the follower's best response. After solving this set of LPs, the solution of the LP with the highest objective value will be optimal overall. Since the follower's best response will always be a pure strategy, we will shorten  $\sigma_1$  to  $\sigma$  for a mixed strategy for the leader and will use  $a_l \in A_l$  to denote the action set of the leader. Letting  $U_l$  and  $U_f$  be the leader and follower's utility functions, the LP is as follows:

General LP

$$\text{maximize } \sum_{a_l} p_{a_l} U_l(a_l, a_f^*)$$

subject to:

$$\forall a_f \in A_f : \sum_{a_l} p_{a_l} U_f(a_l, a_f) \leq \sum_{a_l} p_{a_l} U_f(a_l, a_f^*)$$

$$\sum_{a_l} p_{a_l} = 1$$

In fact, Stackelberg strategies are a generalization of minimax strategies in two-player zero-sum games. As we noted above, computing minimax strategies is equivalent to linear programming, which implies that a linear programming solution for computing Stackelberg strategies is the best that we can hope for computationally. Of course, Nash equilibrium is an alternative generalization of minimax strategies. Recall that the complexity of finding a Nash equilibrium is much worse, as finding any Nash equilibrium is PPAD-complete. Stackelberg strategies also have the significant advantage that they avoid the equilibrium selection problem: there is an optimal value of the game for the leader, which in general corresponds to a single optimal strategy (though not in degenerate cases).

A Stackelberg security game, proposed in Kiekintveld et al. (2009), consists of two players, the leader (defender) and the follower (attacker), and a set of possible targets  $T$  ( $|T| = n$ ). The leader can decide upon a randomized policy of defending the targets, generally with limited defense resources. Many of the security games applications discussed

in Section 1.3 involve the allocation of multiple security resources to multiple targets (or, more generally, a single resource can sometimes be assigned to multiple targets simultaneously, in which case the subset is referred to as a *schedule*, where  $s \in S \subseteq 2^T$ ). Defender resources (denoted by  $\omega \in \Omega$ ) can be either homogeneous, meaning that any defender resource can cover any schedule, or heterogeneous, meaning that, for each resource  $\omega$ , there is a set of schedules  $A(\omega)$  that that resource can cover. A target  $t$  is said to be *covered* if a resource has been assigned to a schedule that covers it. Finally, the utility of the defender (attacker) is denoted as  $U_d^c(t)$  ( $U_a^c(t)$ ) if a target is covered and  $U_d^u(t)$  ( $U_a^u(t)$ ) if it is not covered. It is commonly assumed that  $U_d^c(t) \geq U_d^u(t)$  and  $U_a^c(t) \leq U_a^u(t)$ . Upon observing the leader’s randomized policy (but not the realized defense actions), the follower chooses a target so as to maximize its expected utility.

### 1.3 Related work

The applications discussed at the start of this chapter have driven a wide variety of theoretical and experimental work. However, the basic formulation given above makes a number of assumptions that likely do not hold in the real world. The first assumption that has been questioned is that of observability. While the argument is often made that the follower can perform surveillance to learn the commitment strategy of the leader, the effectiveness of this surveillance is uncertain. Korzhyk et al. (2011c) consider the possibility of this surveillance completely failing, resulting in a simultaneous-move game when this happens. They allay this concern by showing that under a natural restriction on security games, any Stackelberg strategy is also a Nash equilibrium. They also propose an extensive-form game model that explicitly models the defender’s uncertainty about the attacker’s observational capacity. Korzhyk et al. (2011b) give an iterative algorithm for this model and evaluate it empirically.

However, surveillance may only be partially successful or the adversary may not be

completely rational. Pita et al. (2010) model adversary response uncertainty and adversary reward uncertainty in a different fashion, and show how to solve the problem under these uncertainties via modification of the standard mixed-integer linear program (MILP) formulation. In addition, they evaluate these modifications with experiments involving human test subjects using a pirate-and-treasure themed game based on the ARMOR security domain. An additional way to relax the assumption of complete rationality is by modeling bounded rationality via methods such as quantal response. Yang et al. (2012) present algorithms to deal with the computational issues that arise. In addition, they also evaluate these algorithms with human test subjects.

One additional way of handling uncertainty about the attacker is modeling the problem as a Bayesian Stackelberg game. In a Bayesian Stackelberg game, the follower has a type from a set of types, which, together with the actions taken, determines his utility. For Bayesian games with a known prior, the problem of computing the optimal mixed leader strategy is known to be NP-hard [13]. This strategy can be found using a mixed integer program [47] known as DOBBS. While DOBBS is more compact than the multiple-LP approach proposed in [13], it still has significant scalability issues and cannot be used in many settings. Jain et al. (2011) give a technique based on decomposing the Bayesian Stackelberg problem into many smaller restricted problems and show experimentally how this technique helps with scalability. We revisit Bayesian Stackelberg games in Chapter 2. Under the assumption that the targets are completely independent –that is, player utilities only depend on the target attacked and its security configuration– a much more compact representation can be given [25]. Korzhyk et al. (2010) discuss some settings where this formulation is guaranteed to work. We discuss this line of work in more detail in Chapter 5.

Finally, the assumption that the attacker is able to only attack a single target is relaxed in Korzhyk et al. (2011a). They show that in this setting, Nash and Stackelberg strategies for the defender truly diverge. They give a polynomial time algorithm that finds a Nash equilibrium and show that Stackelberg strategies are NP-hard to compute in this context.

## 1.4 Contributions

In Chapters 2-4, we examine computational aspects of computing Stackelberg equilibrium in three types of games: Bayesian games, extensive-form games, and stochastic games. Below is a detailed description of this part's organization:

- Chapter 2 [38]: We study the relationship between different modeling assumptions and the computational tractability and approximability of finding an optimal Stackelberg equilibrium in a setting with multiple follower types. We show that for a finite number of types and a known type distribution, there exists a (number of types) polynomial-time approximation algorithm, and that this approximation is tight unless  $P = NP$  (Section 2.1). We also show that for a finite number of types and adversarial (worst-case) type selection, the problem becomes inapproximable (Section 2.1). Finally, we give two possible representations for a continuous set of follower types, and show that the problem is NP-hard in one case (Section 2.3), and completely inapproximable in a slightly more general case (Section 2.4).
- Chapter 3 [34]: In this chapter, we study commitment in the more general setting of extensive-form games. We first describe in detail a number of different modeling decisions that can be made when modeling a game in the extensive form. We then show how these modeling decisions affect the computational hardness of the problem. We first give polynomial time algorithms for four cases in Section 3.1. Next, we show that when we relax the assumptions in these four cases, the problem usually becomes NP-hard. We show six cases where the problem is NP-hard in Section 3.2.
- Chapter 4 [39]: In this chapter, we unite two lines of research, the computation of optimal strategies to commit to (Stackelberg strategies), and the computation of correlated equilibria of stochastic games by studying the computation of Stackelberg strategies in stochastic games. We provide theoretical results on the value of be-

ing able to commit and the value of being able to correlate (Section 4.3), as well as complexity results about computing Stackelberg strategies in stochastic games (Section 4.4). We then show how to modify the QPACE algorithm [41] to compute approximately optimal Stackelberg strategies (Section 4.5).

In Chapters 5-7 we restrict our attention to a set of games known as Stackelberg security games. Recall that Stackelberg security games impose tight restrictions on the utilities of the two players by forcing a more adversarial relationship between the two players. As the coverage level of a target increases, when the follower chooses to attack that target, the leader's expected utility is rising and the followers is falling. Below is a detailed description of this part's organization:

- Chapter 5 [35]: We focus on how the structure of possible schedules affects the computational hardness. Our approach is similar to the approach taken by Korzhyk et al. (2011) who use the Birkhoff-von Neumann (BvN) theorem to show that under certain conditions there is guaranteed to be a valid mixed strategy that can be found via solving for a marginal solution as an intermediate step. It turns out that for one of the problems we consider, we need a generalization of the BvN theorem that was given by Budish et al. (2013) (which we give a brief overview of in Section 5.1). This generalization allows us to characterize a new class of games where a mixed strategy corresponding to the marginal probabilities is guaranteed to exist, and gives us an algorithm for finding it. In another case, we show how to compute optimal marginal probabilities in a specific way, and show how to directly obtain a mixed strategy that corresponds to these marginal probabilities (Section 5.3). We also show that in a number of settings there exists an instance where the optimal marginal solution does not correspond to any mixed strategy. Then, for most of these settings, we also show that the problem of finding an optimal Stackelberg strategy is in fact NP-hard (Section 5.4).

- Chapter 6 [36]: We modify and extend the previous linear programming techniques for Stackelberg security games to allow for an arbitrary set of security configurations (rather than merely to cover a target, or not), as well as to account for both random and targeted failures, and replace hard constraints on defense resources with costs associated with specific security configurations (Section 6.1). We present and justify a crucial assumption on the nature of interdependencies that allows us to use our LP formulations which fundamentally assume independence between targets (Section 6.2.1). We then offer a simple model of interdependencies based on probabilistic failure cascades satisfying this assumption. Our model makes an explicit distinction between an intrinsic and indirect value of assets, the latter being due entirely to interdependencies. This allows an economically meaningful extension of a well-known independent cascade approach to modeling the spread of infectious diseases or ideas. We demonstrate under this model in trees we can compute expected utilities for all targets in linear time (Section 6.2). We extend our model to capture uncertainty about network structure, and experimentally study the impact of such uncertainty (Sections 6.2.5 and 6.5). We show that our approach is both scalable to realistic security settings and offers much better solutions than state-of-the-art alternatives (Section 6.3.2). Finally, we evaluate experimentally the properties of optimal defense configurations in real and generated networks (Section 6.4).
- Chapter 7 [37]: In this chapter, our end goal is to create a highly scalable mathematical programming approach for optimal interdiction of attack plans. We begin by reviewing an integer programming approach for classical and partial satisfaction planning, which ultimately forms the core of our own framework (Section 7.1). We formulate the optimal attack plan interdiction problem as a large-scale integer linear program, and offer several oracle-based approaches for solving it (Section 7.2). We experimentally evaluate these oracle-based approaches (Section 7.3). We then pro-

ceed to generalize the model to capture two kinds of uncertainty: first, uncertainty about the attacker’s capabilities, costs, and goals, and second, uncertainty about whether a particular planning action is executed or not (Section 7.4). Finally, we consider an attacker model in which the attacker is solving an MDP, and show how the corresponding interdiction problem can be formulate and solved (Section 7.4.3).

Finally, in Chapter 8 [40] we analyze *how much* benefit a player can derive from commitment in various types of games, in a quantitative sense that is similar to concepts such as the value of mediation [4] and the price of anarchy [31, 46]. Specifically, we introduce and study the *value of pure commitment* (the benefit of committing to a pure strategy), the *value of mixed commitment* (the benefit of committing to a mixed strategy), and the *mixed vs. pure commitment* ratio (how much can be gained by committing to a mixed strategy rather than a pure one). We first formally define the *VoPC*, *VoMC*, and *MvP* ratios, in Section 8.1. In Section 8.2, as a “warm-up,” we investigate these ratios in normal-form and symmetric normal-form games. We investigate these ratios in extensive-form and security games in Sections 8.3 and 8.4. In Sections 8.5, 8.6, and 8.7, we investigate these ratios in atomic selfish routing games with linear, quadratic,  $k$ -nomial and arbitrary costs.

## Commitment in Bayesian Games

In many cases, we need to be able to model private information that only one agent has access to. While either or both players could have private information, the most commonly studied case in the context of security games is when only the follower has it. Thus, the follower has a set of *types*  $\Theta$  ( $|\Theta| = \tau$ ), which, together with the actions taken, determine her utility, according to a function  $u_f : \Theta \times A_l \times A_f \rightarrow \mathbb{R}$ . For simplicity, we will not consider situations where the leader's utility also depends on the follower's type; this restriction only strengthens our hardness results. We will refer to these as *Bayesian* games; a *normal-form* game is the special case where there is only a single type.

In a traditional Bayesian Stackelberg game the follower's type ( $\theta$ ) is randomly chosen from a known distribution  $P : \Theta \rightarrow [0, 1]$  after the leader commits. Let  $\text{BR}(\theta, \sigma) \in A_f$  denote the action that the follower plays (that is, his best response, with ties broken in favor of the leader) when his type is  $\theta$  and the leader has committed to playing  $\sigma$ . We note that

$$\text{BR}(\theta, \sigma) \in \arg \max_{a_f \in A_f} \sum_{a_l \in A_l} \sigma(a_l) u_f(\theta, a_l, a_f)$$

The BR function also captures the fact that the follower breaks ties in the leader's favor.

Given the follower type  $\theta$ , the leader's expected utility is

$$\sum_{a_l \in A_l} \sigma(a_l) u_l(a_l, \text{BR}(\theta, \sigma))$$

The leader's expected utility for committing to  $\sigma$  is

$$\sum_{\theta \in \Theta} P(\theta) \sum_{a_l \in A_l} \sigma(a_l) u_l(a_l, \text{BR}(\theta, \sigma))$$

When we take a worst-case perspective, we are interested in a setting with types but without a prior distribution over them (also known as a *pre-Bayesian* game). When we consider games with more than two players,  $A_1$  will be equivalent to  $A_l$  (the set of leader actions), while  $A_2, \dots$  will be the sets of follower actions.

The following aspects of the model will remain the same throughout this chapter.

- We consider two-player, general-sum games that have more than one follower type.
- The leader's utility does not depend *directly* on the follower's type (but it does depend on the follower's action, which can be affected by the follower's type).
- The follower's utility function  $U_f(\theta, a_l, a_f)$  is common knowledge.

We consider two modeling decisions. The first decision concerns whether the type space is discrete or continuous. For the discrete case, we assume that we have a finite number of types, which are explicitly listed. For the continuous case, we assume that the space of possible types is defined by a lower bound and an upper bound for the follower's utility for each action profile  $(a_l, a_f)$ ; every follower payoff matrix that is consistent with these bounds corresponds to some type.

The second modeling decision is whether the follower type is chosen according to a Bayesian model or an adversarial (worst-case) model. Note that the "adversary" is *not* one of the players of the game, in particular, the adversary and the follower are different.

This chapter is laid out as follows. We show that for a finite number of types and a known type distribution, there exists a (number of types) approximation algorithm, and this approximation is tight (Section 2.1). We also show that for a finite number of types and an adversarial (worst-case) type distribution, the problem becomes inapproximable (Section 2.1). Finally, we give two possible representations for a continuous set of follower types, and show that the problem is NP-hard in one case (Section 2.3), and completely inapproximable in a slightly more general case (Section 2.4). This chapter is based on Letchford et al. (2009).

## 2.1 Computing optimal strategies with finitely many types

In this section we study how to compute the optimal mixed strategy when the follower's type is drawn from a known distribution over finitely many types. We refer to this problem as *Bayesian optimization for finite types (BOFT)*. BOFT is defined as:

- We have a set  $\Theta$  of possible follower types,  $|\Theta| = \tau$ .
- The follower's utility function  $u_f(\theta, a_l, a_f)$  is common knowledge.
- Both the follower's utility function  $u_f(\theta, a_l, a_f)$  and the leader's utility function  $u_l(\theta, a_l, a_f)$  are normalized to lie in  $[0,1]$  for all inputs.
- The prior over follower types  $P(\theta)$  is common knowledge.
- An optimal leader strategy is one that maximizes the leader's expected utility.

This problem was first studied in [13], where it was shown to be NP-hard. It also forms the basis for much of the applied work on computing Stackelberg strategies [22]. However, to the best of our knowledge, the approximability of this problem had not yet been studied. We settle the approximability precisely in this subsection.

**Theorem 1.** *For all constant  $\varepsilon > 0$ , no polynomial-time factor- $\tau^{1-\varepsilon}$  approximation exists for BOFT unless  $NP = P$ , even if there are only two follower actions.*

*Proof.* It is known that no polynomial-time factor- $|V|^{1-\varepsilon}$  approximation exists for MAX-INDEPENDENT-SET (given by a graph  $(V, E)$ ), unless  $NP = P$  [69]. We show our result by reducing an arbitrary instance  $(V, E)$  of this problem to a game as follows. For every  $v \in V$ , there is a follower type  $\theta^v$ , and a leader action  $a_l^v$ . The prior over follower types is uniform. There are two follower actions,  $A$  and  $B$  (for each follower type). The leader gets utility 1 if the follower plays  $A$ , and 0 otherwise. The follower's utility is defined as follows.

- For all  $v \in V$ ,  $u_f(\theta^v, a_l^v, A) = |V|$ .
- For all  $v, w \in V$  with  $v \neq w$ ,  $u_f(\theta^v, a_l^w, A) = 0$ .
- For all  $(v, w) \in E$ ,  $u_f(\theta^v, a_l^w, B) = 1 + |V|^2$
- For all  $(v, w) \notin E$ ,  $u_f(\theta^v, a_l^w, B) = 1$ .

Suppose there is an independent set  $S$  of size  $k$  in  $(V, E)$ . Consider a mixed strategy that places probability  $\frac{1}{k}$  on each  $a_l^v$  with  $v \in S$ . Then, for every type  $\theta^v$  with  $v \in S$ , the follower will play  $A$ , because the follower will get  $\frac{n}{k} \geq 1$  for playing  $A$ , and 1 for playing  $B$  (because no  $a_l^w$  with  $(v, w) \in E$  is ever played, because  $S$  is an independent set).

Correspondingly, suppose there is a leader strategy that gets  $k$  types to play  $A$ . Let  $S$  be the set of vertices  $v$  such that the follower plays  $A$  for  $\theta^v$ ; we will show it is an independent set. If  $\theta^v$  plays  $A$ , then  $a_l^v$  must get probability at least  $1/n$  (to make playing  $A$  optimal for the follower). But, no action  $a_l^w$  with  $(v, w) \in E$  can get probability at least  $\frac{1}{n}$ , because in that case the expected utility for the follower (with type  $\theta^v$ ) of playing  $B$  is at least  $(\frac{1}{n})(1 + n^2) > n$ . So the  $k$  types must constitute an independent set.

Hence, we have shown that the number of types playing  $A$  (which is proportional to the leader's utility) for the optimal leader strategy is equal to the size of the maximum

independent set. Since the number  $\tau$  of types for the follower is equal to  $|V|$ , this gives us the desired result.  $\square$

**Theorem 2.** *There is a polynomial-time factor- $\tau$  approximation algorithm for BOFT.*

*Proof.* Let  $\sigma^*$  be an optimal leader strategy, that is,

$\sigma^* \in \arg \max_{\sigma} \sum_{\theta \in \Theta} P(\theta) \sum_{a_l} \sigma(a_l) u_l(a_l, BR_f(\theta, \sigma))$ . Consider the following simple randomized algorithm: choose a type  $\theta$  uniformly at random and play a mixed leader strategy  $\sigma^\theta$  that maximizes utility against that single type, that is,

$\sigma^\theta \in \arg \max_{\sigma} \sum_{a_l} \sigma(a_l) u_l(a_l, BR_f(\theta, \sigma))$ . (We can find such a mixed leader strategy in polynomial time by the linear programming approach from [13].) For every  $\theta$ , we have

$\sum_{a_l} \sigma^\theta(a_l) u_l(a_l, BR_f(\theta, \sigma^\theta)) \geq \sum_{a_l} \sigma^*(a_l) u_l(a_l, BR_f(\theta, \sigma^*))$ . The probability that  $\theta$  is chosen both by our algorithm and by nature as the type of the follower is  $(1/\tau)P(\theta)$ . Because utilities are bounded below by zero, the expected utility that we receive is at least

$$\sum_{\theta \in \Theta} (1/\tau) P(\theta) \sum_{a_l} \sigma^\theta(a_l) u_l(a_l, BR_f(\theta, \sigma^\theta)) \geq$$

$(1/\tau) \sum_{\theta \in \Theta} P(\theta) \sum_{a_l} \sigma^*(a_l) u_l(a_l, BR_f(\theta, \sigma^*))$ . Hence, this randomized algorithm results in a factor- $\tau$  approximation. (We emphasize that this algorithm randomly chooses a mixed strategy to commit to, which is not the same as committing to the corresponding mixture of those mixed strategies.)

Instead of randomizing uniformly over which of the  $\sigma^\theta$  to commit to, we can instead, for each  $\theta$ , evaluate the total expected utility that results from committing to the strategy  $\theta$  (which is

$$\sum_{\theta' \in \Theta} P(\theta') \sum_{a_l} \sigma^\theta(a_l) u_l(a_l, BR_f(\theta', \sigma^\theta))),$$

and choose one that maximizes this expected utility. This cannot lead to a lower expected utility for the leader; hence, this gives us a deterministic algorithm with the same approximation guarantee.  $\square$

## 2.2 Computing worst-case optimal strategies with finitely many types

A prior distribution over follower types is not always readily available. In that case, we may wish to optimize for the worst-case type (equivalently, the worst-case distribution over types). We note that the worst-case type depends on the mixed strategy that we choose, so that this is not the same problem as optimizing against a single type. We refer to this problem as *worst-case optimization for finite types (WOFT)*:

- We have a set  $\Theta$  of possible follower types,  $|\Theta| = \tau$ .
- The follower's utility function  $u_f(\theta, a_l, a_f)$  is common knowledge.
- An optimal leader strategy is one that maximizes the worst-case expected utility for the leader, where the worst case is taken over follower types (but we are taking the expectation over the mixed strategy). That is, an adversary (not equal to the follower) chooses the follower type after the leader mixed strategy is chosen, but before the pure-strategy realization.

It turns out that WOFT is even less approximable than BOFT.

**Theorem 3.** *WOFT is completely inapproximable in polynomial time, unless  $P=NP$  (that is, it is hard to distinguish between instances where the leader can get at least 1 in the worst case, and instances where the leader can only get 0)—even if there are only four follower actions.*

*Proof.* We reduce an arbitrary instance of 3SAT to a game such that the leader can obtain an expected utility of 1 if the 3SAT instance is satisfiable, and 0 otherwise. The 3SAT instance consists of  $n$  variables,  $x_1, \dots, x_n$ , and  $m$  clauses,  $C_1, \dots, C_m$ . We create one type for each variable and for each clause. In the game, for every variable  $x_i$ , we have two leader actions,  $a_l^{+x_i}$  and  $a_l^{-x_i}$ . The follower has four actions,  $A, B, C$ , and  $D$ . The leader

gets utility 0 if the follower plays A, and 1 otherwise. There are two kinds of follower type: one for each variable ( $\theta^{x_i}$ ) and one for each clause ( $\theta^{C_j}$ ).

For a type  $\theta^{x_i}$  corresponding to a variable, we make it so that actions C and D are always suboptimal for the follower, so we only consider A and B. We let  $u_f(\theta^{x_i}, a_l^{+x_i}, A) = u_f(\theta^{x_i}, a_l^{-x_i}, A) = n$ , and  $u_f(\theta^{x_i}, a_l^{+x_j}, A) = u_f(\theta^{x_i}, a_l^{-x_j}, A) = 0$  for  $i \neq j$ . Also, we let  $u_f(\theta^{x_i}, a_l, B) = 1$  for all  $a_l$ . The following table gives the payoff matrix for type  $\theta^{x_1}$  as an example.

$$M_{x_1}: \begin{array}{c|cc} & A & B \\ \hline +x_1 & 0,n & 1,1 \\ \hline -x_1 & 0,n & 1,1 \\ \hline & 0,0 & 1,1 \\ \hline & 0,0 & 1,1 \\ \hline & \cdot & \cdot \\ \hline & \cdot & \cdot \\ \hline & \cdot & \cdot \end{array}$$

For a type  $\theta^{C_j}$  corresponding to a clause, for each literal  $\lambda$  in the clause (where the set of all literals is  $\{+x_i : i \in \{1, \dots, n\}\} \cup \{-x_i : i \in \{1, \dots, n\}\}$ ), exactly one of the three actions B, C, D will give the follower utility  $n$  if the leader plays  $a_l^\lambda$  (and each of these three actions will correspond to one of the three literals in the clause). Playing A always gives the follower utility 1. Otherwise, the follower gets 0. For example, if  $C_1 = (+x_1 \vee -x_2 \vee +x_4)$ , then the following table gives the payoff matrix for type  $\theta^{C_1}$ .

$$M_{c_1}: \begin{array}{c|ccccc} & A & B & C & D \\ \hline +x_1 & 0,1 & 1,n & 1,0 & 1,0 \\ \hline -x_1 & 0,1 & 1,0 & 1,0 & 1,0 \\ \hline +x_2 & 0,1 & 1,0 & 1,0 & 1,0 \\ \hline -x_2 & 0,1 & 1,0 & 1,n & 1,0 \\ \hline +x_3 & 0,1 & 1,0 & 1,0 & 1,0 \\ \hline -x_3 & 0,1 & 1,0 & 1,0 & 1,0 \\ \hline +x_4 & 0,1 & 1,0 & 1,0 & 1,n \\ \hline -x_4 & 0,1 & 1,0 & 1,0 & 1,0 \end{array}$$

We now show that the leader can obtain a utility of 1 in this game against every type if

and only if the 3SAT instance has a satisfying assignment (and will get 0 against at least one type otherwise). Let  $\sigma$  be the mixed strategy to which the leader commits.

For each variable  $x_i$ , if  $\sigma(a_i^{+x_i}) + \sigma(a_i^{-x_i}) \leq \frac{1}{n}$ , then a follower of type  $\theta^{x_i}$  will play  $B$ , otherwise it will play  $A$ . Hence, the leader will get 1 for all types corresponding to variables if and only if the above inequality holds for every variable  $x_i$ ; otherwise, the leader will obtain 0 against at least one type corresponding to a variable.

For each clause  $C_j$ , if for at least one of the three literals  $\lambda$  in the clause, we have  $\sigma(a_l^\lambda) \geq \frac{1}{n}$ , then a follower of type  $\theta^{C_j}$  will play  $B$ ,  $C$ , or  $D$ ; otherwise, it will play  $A$ . Hence, the leader will get 1 for all types corresponding to clauses if and only if every clause contains at least one literal for which the above inequality holds; otherwise, the leader will obtain 0 against at least one type corresponding to a clause.

Now, suppose that the 3SAT instance has a satisfying assignment. Then, consider the mixed strategy that places probability  $1/n$  on every literal that is set to *true* in the satisfying assignment. For every variable  $x_i$ , we have  $\sigma(a_i^{+x_i}) + \sigma(a_i^{-x_i}) \leq \frac{1}{n}$ , so the follower will play  $B$  for all the types corresponding to variables. For every clause, for at least one of the three literals  $\lambda$  in the clause, we have  $\sigma(a_l^\lambda) \geq \frac{1}{n}$  (because the assignment satisfies the formula), so the follower will play  $B$ ,  $C$ , or  $D$  for all the types corresponding to clauses. Hence, the leader obtains utility 1 for every follower type.

Conversely, suppose that there exists a mixed strategy such that the leader obtains positive utility for every follower type. For every variable  $x_i$ , we must have  $\sigma(a_i^{+x_i}) + \sigma(a_i^{-x_i}) \leq \frac{1}{n}$ . Hence, at most one of  $a_i^{+x_i}$  and  $a_i^{-x_i}$  can receive probability at least  $1/n$ . Now consider the following assignment: if  $a_i^{+x_i}$  receives probability at least  $1/n$ , set  $x_i$  to *true*; if  $a_i^{-x_i}$  receives probability at least  $1/n$ , set  $x_i$  to *false*; otherwise, set  $x_i$  arbitrarily. Because the leader receives utility at least 1 against every type corresponding to a clause, for every clause, for at least one of the three literals  $\lambda$  in the clause, we must have  $\sigma(a_l^\lambda) \geq \frac{1}{n}$ . But that means that the clause either contains some  $+x_i$  where  $x_i$  is set to *true*, or some  $-x_i$

where  $x_i$  is set to *false*. It follows that our assignment is a satisfying assignment.

□

### 2.3 Optimizing for the worst type with ranges

So far, we have assumed that the space of possible types is represented by explicitly listing the (finitely many) types and the corresponding utilities. However, this representation of the uncertainty that the leader has over the follower's preferences is not always convenient. For example, the leader may have a rough idea of every follower payoff, which could be represented by a range in which that payoff must lie. This corresponds to a continuous type space for the follower: every setting of all the follower payoffs within the ranges corresponds to a type.

In this subsection, we study the problem of maximizing the leader's worst-case utility over all types (instantiations of the follower payoffs within the ranges). Later in the subsection, we also consider a generalization where the follower payoffs in different entries can be linked to each other.

For example, consider the following game with ranges:

	$L$	$R$
$U$	0, [1,2]	1, 0
$D$	1, 0	0, [1,2]

The leader is unsure about the follower's utility for  $(U, L)$  and  $(D, R)$ , each of which is known to lie somewhere in the range  $[1, 2]$  (they can vary independently). The follower knows his utilities. If the leader places less than  $1/3$  probability on  $U$ , then the follower is guaranteed to play  $R$ ; this results in a utility of at most  $1/3$  for the leader. If the leader places more than  $2/3$  probability on  $U$ , then the follower is guaranteed to play  $L$ ; this results in a utility of at most  $1/3$  for the leader. If the leader places probability between  $1/3$  and  $2/3$  on  $U$ , then the follower may end up playing either  $L$  or  $R$ ; by placing probability  $1/2$  on  $U$ , the leader obtains an expected utility of  $1/2$ , which is optimal.

We refer to this problem as *worst-case optimization for range types (WORT)*:

- For every  $(a_l, a_f)$ , the leader has a range in which the follower utility might lie,  $u_f(a_l, a_f) \in [u_f^l(a_l, a_f), u_f^h(a_l, a_f)]$ . The leader knows her own utilities  $u_l(a_l, a_f)$ .
- An optimal leader strategy is one that maximizes the worst-case expected utility for the leader, where the worst-case values of

**Theorem 4.** *WORT is NP-hard.*

*Proof.* We reduce an arbitrary instance of 3-COVER (where we are given a set of elements  $S$  ( $|S| = n$ ), a collection of subsets  $S_i \subseteq S$  with  $|S_i| = 3$ , and we are asked whether all of  $S$  can be covered with  $n/3$  of the  $S_i$ ) to a game with ranges where the leader can obtain an expected utility of at least  $3/n$  if and only if a 3-cover exists.

For each  $S_i$ , let there be both a row  $a_l^{S_i}$  and a column  $a_f^{S_i}$ . Also, let there be a column  $a_f^s$  for each  $s$  (these columns are really bad for the leader and must be avoided). Let the utilities be defined as follows:

$$u_l(a_l^{S_i}, a_f^{S_i}) = 1$$

$$u_l(a_l^{S_i}, a_f^{S_j}) = 0 \text{ for } i \neq j$$

$$u_f(a_l^{S_i}, a_f^{S_i}) \in [0, 1]$$

$$u_f(a_l^{S_i}, a_f^{S_j}) = 0 \text{ for } i \neq j$$

$$u_f(a_l^{S_i}, a_f^s) = 1 \text{ if } s \notin S_i$$

$$u_f(a_l^{S_i}, a_f^s) = -n \text{ if } s \in S_i$$

If there exists a 3-cover (of size  $n/3$ ), then the leader can obtain guaranteed utility  $3/n$ , as follows: randomize uniformly over the strategies corresponding to the 3-cover (probability  $3/n$  each). The follower will not be incentivized to play any  $a_f^s$ , because that gives him an expected utility of at most  $-3 + 1 = -2$ . The follower will not be incentivized to play an  $a_f^{S_j}$  for which  $S_j$  is not in the 3-cover. because it will give him utility 0 (note

that ties are broken in favor of the leader, as always). The follower may be incentivized to play any  $a_f^{S_i}$  for which  $S_i$  is in the 3-cover; for each of these, the leader will get  $3/n$  in expectation.

Conversely, if the leader can get guaranteed utility  $3/n$ , consider the set of all the  $S_i$  for which  $a_l^{S_i}$  receives positive probability for the leader. The claim is that this must be a 3-cover (of size  $3/n$ ). First, the follower cannot be incentivized to play any  $a_f^s$ . Hence, for each  $s$ , some  $a_l^{S_i}$  with  $s \in S_i$  must get positive probability for the leader. Now suppose strictly more than  $n/3$  of the  $a_l^{S_i}$  get positive probability for the leader. Then one of them must get probability less than  $3/n$ . The column player might be incentivized to play the corresponding  $a_f^{S_i}$  (since that may be the only one that ever gives the follower positive utility), in which case the row player's expectation is less than  $3/n$ , contrary to assumption.  $\square$

## 2.4 Worst-case optimization for linked range types

We now define a generalization of WORT (from subsection 2.3, which we can prove is inapproximable unless  $P = NP$ ). This generalization allows the follower's payoffs to be linked across entries. We refer to this problem as *worst-case optimization for linked range types (WOLRT)*. Specifically, instead of having ranges for each follower entry, we now have a linear expression for each follower entry which may involve *symbols*: an example expression would be  $3c_1 + 4c_2 + 1$ . For each symbol, there is a range (for example,  $c_1 \in [0, 1]$ )—in fact, without loss of generality, we can assume every range is  $[0, 1]$ , and a symbol can occur in multiple entries.

For example, consider the following game with linked ranges, with  $c_1, c_2 \in [0, 1]$ :

	$L$	$R$
$U$	$0, 1 + c_1$	$1, 0$
$D$	$1, 0$	$0, 1 + c_1/2 + c_2/2$

If the leader places less than  $3/7$  probability on  $U$ , then the follower is guaranteed to play  $R$ ; this results in a utility of at most  $3/7$  for the leader. If the leader places more than  $3/5$  probability on  $U$ , then the follower is guaranteed to play  $L$ ; this results in a utility of at most  $2/5$  for the leader. If the leader places probability between  $3/7$  and  $3/5$  on  $U$ , then the follower may end up playing either  $L$  or  $R$ ; by placing probability  $1/2$  on  $U$ , the leader obtains an expected utility of  $1/2$ , which is optimal.

We note that WOLRT generalizes WORT, because we can have a separate symbol for every entry.

**Theorem 5.** *WOLRT is completely inapproximable in polynomial time, unless  $P=NP$  (that is, it is hard to distinguish between instances where the leader can get at least 1 in the worst case, and instances where the leader can only get 0).*

*Proof.* We reduce an arbitrary SET-COVER instance (where we are given a set  $S$ , a collection of subsets  $S_i$  of  $S$ , and a number  $k$ , and are asked whether all of  $S$  can be covered with at most  $k$  of the  $S_i$ ) to a WOLRT instance such that the leader can get utility 1 in the worst case if there is a set cover of size at most  $k$ , and 0 otherwise.

With every  $s \in S$ , we associate a symbol  $c_s \in [0, 1]$ . For every  $S_i$ , the leader has an action  $a_l^{S_i}$ , and the follower has an action  $a_f^{S_i}$ . Additionally, for every  $s \in S$ , the follower has an action  $a_f^s$ . We have:

$$u_l(\cdot, a_f^s) = 0$$

$$u_l(\cdot, a_f^{S_i}) = 1$$

$$u_f(\cdot, a_f^s) = c_s$$

$$u_f(a_l^{S_i}, a_f^{S_i}) = k \sum_{s \in S_i} c_s$$

$$u_f(a_l^{S_i}, a_f^{S_j}) = 0 \text{ for } i \neq j$$

If there is a covering of size  $k$ , then the leader can uniformly randomize over the  $a_l^{S_i}$

corresponding to that covering. Then, for any  $s' \in S$ , if the follower plays one of the  $a_f^{S_i}$  where  $S_i$  is in the covering and  $s' \in S_i$ , his expected utility is at least  $(1/k) \cdot k(\sum_{s \in S_i} c_s) \geq c_{s'}$ ; so there is no reason for the follower to play  $a_f^{s'}$ , and the leader is guaranteed a utility of 1.

Conversely, suppose that there is a strategy  $\sigma$  that guarantees the leader positive utility, that is, it guarantees that the follower will play one of the  $a_f^{S_i}$ . For any  $s' \in S$ , consider the scenario where  $c_{s'} = 1$  and the other  $c_s$  are 0. The follower is incentivized to play some  $a_f^{S_i}$ ; it must be that  $s' \in S_i$ , and the follower's expected payoff for playing this is  $\sigma(a_f^{S_i}) \cdot k \sum_{s \in S_i} c_s = \sigma(a_f^{S_i}) \cdot k$ , so it follows that  $\sigma(a_f^{S_i}) \geq 1/k$ . There can be at most  $k$  subsets  $S_i$  for which this is true, and they must cover all the  $s' \in S$ , so it follows there is a covering of size at most  $k$ .  $\square$

## 2.5 Mixed integer program formulations of BOFT and WOFT

First let us introduce a mixed integer program (which, in our view, simplifies the known mixed integer program [47] slightly, but the idea is similar). It uses auxiliary variables  $q(\theta, a_l, a_f)$ , which correspond to the probability that  $a_l, a_f$  are played, given that the follower has type  $\theta$ —which will be equal to 0 if  $a_f$  is not a best response for  $\theta$ , and equal to  $\sigma(a_l)$  otherwise. It also uses binary indicator variables  $b(\theta, a_f) \in \{0, 1\}$  for whether the best response for type  $\theta$  is  $a_f$ .

<p><b>maximize</b> <math>\sum_{\theta} P(\theta) \sum_{a_l, a_f} q(\theta, a_l, a_f) u_l(a_l, a_f)</math>  <b>subject to</b>  <math>(\forall \theta) \sum_{a_f} b(\theta, a_f) = 1</math>  <math>(\forall \theta, a_l, a_f) q(\theta, a_l, a_f) \leq b(\theta, a_f)</math>  <math>(\forall \theta, a_l) \sum_{a_f} q(\theta, a_l, a_f) = \sigma(a_l)</math>  <math>(\forall \theta, a_f, a'_f) \sum_{a_l} q(\theta, a_l, a_f) (u_f(\theta, a_l, a_f) - u_f(\theta, a_l, a'_f)) \geq 0</math>  <math>\sum_{a_l} \sigma(a_l) = 1</math>  <math>(\forall a_l) \sigma(a_l) \geq 0</math></p>
--

The following is a mixed integer program (MIP) formulation for WOFT. In this MIP, we assume that all payoffs are normalized to lie in  $[0, 1]$ . Again, we use a binary variable

$b(\theta, a_f) \in \{0, 1\}$  that indicates whether  $a_f$  is the best response for  $\theta$ . We also use variables  $U_l$  (the leader's worst-case utility),  $U_l(\theta)$  (the leader's utility if the type is  $\theta$ ),  $U_f(\theta, a_f)$  (the follower's utility for playing  $a_f$  given  $\theta$ ),  $U'_f(\theta, a_f)$  (equal to  $U_f(\theta, a_f)$  if  $a_f$  is the best response for  $\theta$ , 0 otherwise),  $U_f(\theta)$  (the follower's utility if the type is  $\theta$ ).

<p><b>maximize</b> <math>U_l</math>  <b>subject to</b>  <math>(\forall \theta) \sum_{a_f} b(\theta, a_f) = 1</math>  <math>(\forall \theta, a_f) U_f(\theta, a_f) = \sum_{a_l} \sigma(a_l) u_f(\theta, a_l, a_f)</math>  <math>(\forall \theta, a_f) U'_f(\theta, a_f) \leq U_f(\theta, a_f)</math>  <math>(\forall \theta, a_f) U'_f(\theta, a_f) \leq b(\theta, a_f)</math>  <math>(\forall \theta) U_f(\theta) = \sum_{a_f} U'_f(\theta, a_f)</math>  <math>(\forall \theta, a_f) U_f(\theta) \geq U_f(\theta, a_f)</math>  <math>(\forall \theta, a_f) U_l(\theta) \leq \sum_{a_l} \sigma(a_l) u_l(a_l, a_f) + (1 - b(\theta, a_f))</math>  <math>(\forall \theta) U_l \leq U_l(\theta)</math>  <math>\sum_{a_l} \sigma(a_l) = 1</math>  <math>(\forall a_l) \sigma(a_l) \geq 0</math></p>
--

## Commitment in Extensive-Form Games

In this chapter, we examine how hard it is to solve for a Stackelberg (behavioral) strategy in an extensive-form game under various restrictions. An extensive-form game consists of a rooted tree with additional information. Each leaf node of the tree specifies the payoffs for all players. Each internal node of the tree is associated with a player, who acts at that node, resulting in a move to one of that node's children. A player's nodes are partitioned into *information sets*, and the player cannot distinguish among nodes in an information set (this implies that the player has the same set of possible actions at every node in the same information set). Let  $m$  be the number of leaf nodes in the tree, and  $n$  the number of internal nodes. We use  $\sigma$  here to represent commitment to *behavioral strategies*: a behavioral strategy for a player associates with each information set for that player a probability distribution over the actions associated with that information set. A behavioral strategy is *pure* if all of its probabilities are 0 or 1; in general, a behavioral strategy is *mixed*. Our results here focus on how the following modeling decisions affect the computational tractability of solving for the optimal (leader utility maximizing) Stackelberg strategy in extensive form games.

- *Chance nodes (moves by Nature)*. Does the game include moves by Nature? (Nature is a player that plays according to a fixed behavioral strategy and has no stake in the game; when we count the number of players, Nature is not included.)
- *Commitment to pure strategies vs. mixed strategies*. Is the leader able to commit to a mixed behavioral strategy, or only to a pure one?
- *Tree or DAG*. Conceptually, in extensive-form games, there is no loss of generality in assuming that the game is represented as a tree: if two different paths of play both lead us to the same state of the game, it is always possible to simply have two copies of that state, corresponding to the different paths that may have led us there. However, this duplication can result in an exponential blowup in representation size. Moreover, when we consider commitment to behavioral strategies, it may not be reasonable to allow a player to commit to one action at one node, and another at a different node that in reality represents the same state. For this reason, we consider not only the standard representation of an extensive-form game using a tree (and additional information), but also using a directed acyclic graph (DAG). When a game is represented using a DAG, the probabilities to which a player commits at a node cannot depend on the path taken to that node. Figure 3.1 shows an example a modified form of matching pennies where the winner has a chance of winning an additional unit of utility. This game is depicted as a DAG in two different equivalent forms (in the second form, when two nodes are both shown as parents of a subtree, for example  $P_1$ , this means that they are both parents of a single copy of that subtree). The second form will be how DAG's will be drawn for the rest of this chapter.
- *Number of players*. Some of the variants will turn out to become hard at two players, some at three players, and others are easy for any number of players.

- *Restricted or costly commitment.* It may be the case that the leader is able to commit only at some nodes. Alternatively, it may be the case that for each node, there is a non-negative cost associated with committing at that node, which will be subtracted from the leader's utility at the end of the game. (Restricted commitment is the special case where all costs are 0 or  $\infty$ ; hence, a positive result for costs implies a positive result for restrictions, and vice versa for negative results.)
- *Perfect or imperfect information.* A game has perfect information if all information sets have size 1.

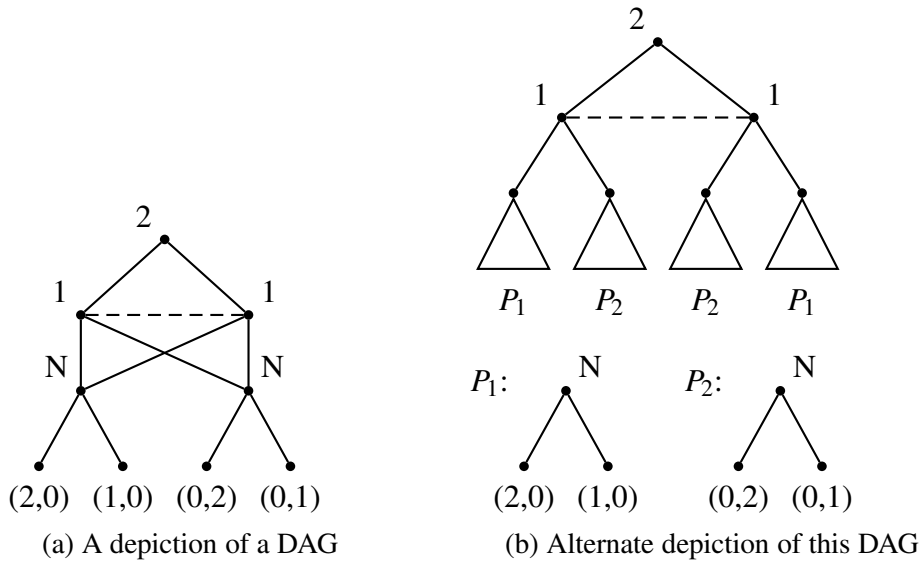


FIGURE 3.1: Example of DAG depiction methods.

One issue that deserves some discussion is the following: does the leader commit at every information set, or can the leader selectively choose to commit at some, but not other, information sets? In many cases, it will be without loss of generality to assume that the leader commits at every information set (assuming that there are no restrictions or costs to doing so). To argue this, we first need to consider what happens when the leader does not commit everywhere. After the commitment phase, an extensive-form game

results, where every node at which the leader committed is replaced by a move by Nature, in accordance with the mixture to which the leader committed there. We assume that some Nash equilibrium of this modified game will result; since we are only interested in games of perfect recall, we can assume that the equilibrium is in behavioral strategies [32]. Moreover, we assume that if the extensive-form game is one of perfect information, then a *pure* Nash equilibrium will be played.

With these assumptions, if commitment to mixed behavioral strategies is possible, then the leader may as well commit to the probabilities prescribed by the Nash equilibrium (of the modified game) that is best for her; assuming ties are broken in the leader's favor, she will do no worse.

Even if commitment to mixed behavioral strategies is not allowed, then still, if the extensive-form game is one of perfect information, since we assume that a pure Nash equilibrium will be played, the leader can simply commit to the actions prescribed by the best such equilibrium for her.

Nevertheless, in games of imperfect information where commitment to mixed strategies is not allowed, we must admit that it can be optimal to not commit everywhere (for example, matching pennies; it is also easy to construct examples where the leader wants to commit at some, but not all, information sets). Our hardness result here assumes that the leader can choose to commit at a subset of the information sets.

Throughout, if there are multiple equilibria in the game that results after the commitment stage, we assume that one of the equilibria that is optimal for player 1 will be played (with the caveat that if the game has perfect information, we assume that the best *pure* equilibrium for player 1 will be played). This is consistent with the common assumption that the follower breaks ties in the leader's favor.

A summary of our results appears in Figure 3.2, in the form of a decision tree. (This is the only tree in this chapter that does not correspond to a game.) This chapter is based on Letchford and Conitzer (2010).

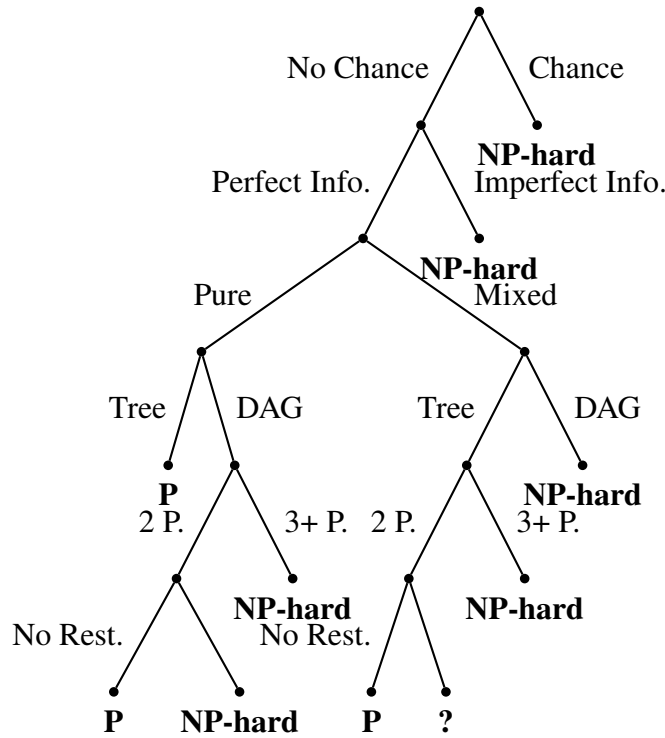


FIGURE 3.2: Summary of extensive form results. At any node, some of the aspects of the problem are fixed, others are not. Every leaf node states whether the problem is in P or NP-hard at that leaf; this result then applies for *any* way of fixing the remaining aspects of the problem.

### 3.1 Positive results

In this section we give polynomial-time algorithms for the following problems:

Case 1. Perfect-information games in tree form with no chance nodes and no costs/restrictions on commitment, where we allow for any number of players, but restrict the problem to pure strategy commitment (Theorem 6).

Case 2. Perfect-information games in tree form with costs/restrictions on commitment with no chance nodes, where we allow for any number of players, but restrict the problem to pure strategy commitment (Theorem 7).

Case 3. Perfect-information games in DAG form with no chance nodes and no costs/restrictions on commitment, where we allow for any number of players, but restrict the problem to pure strategy commitment (Theorem 8).

Case 4. Perfect-information games that allow for mixed strategy commitment in tree form with no chance nodes and no costs/restrictions on commitment, with any number of players (Theorem 9).

### 3.1.1 Case 1. No chance, pure strategies, trees, no costs or restrictions, perfect information

**Theorem 6.** *In perfect-information games in tree form with no chance nodes and no costs/restrictions on commitment, the optimal pure strategy to commit to can be found in  $O(nm)$  time (for any number of players).*

*Proof.* The algorithm contains two main steps. The first step will be a bottom-up dynamic program for determining, for each node  $v$ , a set  $S_v$ , which is the subset of  $v$ 's descendant leaf nodes that can be achieved by an appropriate commitment by player 1. We then choose the best outcome  $l^* \in S_r$  for player 1, where  $r$  is the root of the tree. Then, in the second (top-down) stage, we determine the appropriate strategy to commit to to achieve  $l^*$ .

#### Dynamic programming step (upward pass)

For each leaf node  $l$ ,  $S_l = \{l\}$ . For every internal node  $v$  at which player 1 takes an action, 1 can achieve any leaf that can be obtained from any of its children, by committing to the corresponding child and committing appropriately from there on. Hence,  $S_v = \bigcup_{w \text{ child of } v} S_w$ .

The case of an internal node  $v$  at which player  $i \neq 1$  takes an action is slightly more complicated. Here, it will be true that  $S_v \subseteq \bigcup_{w \text{ child of } v} S_w$ , but in general not all elements of this union will be included, because it may not be feasible to incentivize player  $i$  to choose every one of them. For each child  $w$  of  $v$ , let  $m(w) \in \arg \min_{l \in S_w} u_i(l)$ ; this leaf corresponds to the most we can “punish”  $i$  for going to  $w$ . Then,  $S_v = \bigcup_{w \text{ child of } v} \{l \in S_w : \max_{w' \neq w} u_i(m(w')) \leq u_i(l)\}$ .

These recursive equalities allow us to use dynamic programming to compute  $S_r$ ; we can then choose  $l^* \in \arg \max_{l \in S_r} u_1(l)$ .

### Stackelberg strategy determination (downward pass)

We need a procedure  $\text{strategy}(v, l)$  that, given a node  $v$  and one of its descendant leaves  $l \in S_v$ , specifies how to commit in the subtree rooted at  $v$  so that the outcome  $l$  results. If player 1 acts at  $v$ , then let  $w$  be the child of  $v$  that is an ancestor of  $l$ ; simply commit to going to  $w$  and recursively call  $\text{strategy}(w, l)$ . (It does not matter how 1 commits at other children of  $v$  since they will not be reached.) If player  $i \neq 1$  acts at  $v$ , then let  $w$  be the child of  $v$  that is an ancestor of  $l$ . For  $w$ , recursively call  $\text{strategy}(w, l)$ . For all other children  $w' \neq w$  of  $v$ , recursively call  $\text{strategy}(w', m(w'))$ .

### Runtime analysis

We recall that there are  $m$  leaves and  $n$  internal nodes. Each time we reach an internal node, we require a constant number of operations on a subset of the leaves; each of these operations requires at most linear time; we reach every internal node twice (once on the upward and once on the downward pass).  $\square$

#### 3.1.2 Case 2. No chance, pure strategies, trees, perfect information

In this subsection, we extend Theorem 6 to the case where commitment at an information set can come at a cost (or where there are restrictions on commitment at some information sets, which is the special case where the cost is  $\infty$ ). This comes at the cost of an additional factor  $m$  in the runtime.

**Theorem 7.** *In a perfect-information games in tree form with no chance nodes, the optimal pure strategy to commit to can be found in  $O(nm^2)$  time (with any number of players, and with costs or restrictions on commitment).*

*Proof.* Each node  $v$  now has a cost of commitment associated with it,  $c_v \geq 0$ . The algorithm contains two main steps. The first step will be a bottom-up dynamic program for determining, for each node  $v$ , a set  $S_v$ , which is the subset of  $v$ 's descendant leaf nodes that can be achieved by an appropriate commitment by player 1. Additionally, we will calcu-

late at every node  $v$ , for every leaf node  $l \in S_v$ , the lowest cost  $c_v(l)$  required to achieve an outcome of  $l$  at node  $v$ . Then, at the root  $r$ , we choose the best outcome  $l^* \in S_r$  for player 1, where  $l^* \in \arg \max_{l \in S_r} \{u_1(l) - c_r(l)\}$ . In the second (top-down) stage, we determine the appropriate strategy to commit to to achieve  $l^*$ .

### **Dynamic programming step (upward pass)**

For each leaf node  $l$ ,  $S_l = \{l\}$  and  $c_l(l) = 0$ . For every internal node  $v$  at which player 1 takes an action, 1 can achieve any leaf that can be obtained from any of its children, by committing to the corresponding child and committing appropriately from there on. Hence,  $S_v = \bigcup_{w \text{ child of } v} S_w$ . However, for a given leaf node  $l$ , it may be possible to achieve  $l$  without committing at  $v$ . Thus, in order to compute  $c_v(l)$ , we calculate  $c_v^1(l)$ , which is the minimum cost to achieve  $l$  by committing at  $v$ , and  $c_v^2(l)$ , which is the minimum cost to achieve  $l$  by not committing at  $v$  (but potentially committing further down in the tree). We may have  $c_v^2(l) = \infty$  if it is impossible to achieve  $l$  from  $v$  without committing at  $v$ . We then have that  $c_v(l) = \min(c_v^1(l), c_v^2(l))$ .

In either calculation,  $l$  must be selected in the child  $w$  that it descends from, so that  $c_v(l)$  will always include  $c_w(l)$ . Where the two calculations diverge is as follows. If we commit at  $c_v$ , we pay a cost of  $c_v$  for committing at  $v$ , but we will not need to make any commitments in any of  $v$ 's other children. Thus,  $c_v^1(l) = c_w(l) + c_v$ . To achieve  $l$  without commitment at  $v$ , the outcomes in every other child  $w' \neq w$  must be such that  $u_1(w') \leq u_1(l)$ . The lowest-cost leaf node  $l_v^{w'}$  that achieves this for  $w' \neq w$  is  $l_v^{w'} \in \arg \min_{(l' \in S_{w'} \text{ and } u_1(l') \leq u_1(l))} c_{w'}(l')$ , and its cost is  $c_{w'}(l_v^{w'})$ . We note that  $l_v^{w'}$  may not be well defined, because it may be that for all  $l' \in S_{w'}$ ,  $u_1(l') > u_1(l)$ , so that min is taken over an empty set. This corresponds to the case where it is impossible to make player 1 prefer  $l$  over  $w'$  (without commitment at  $v$ ); in this case we define  $c_{w'}(l_v^{w'}) = \infty$ . Then, the minimal cost for reaching  $l$  from  $v$  without commitment at  $v$  is  $c_v^2(l) = c_w(l) + \sum_{w' \neq w} c_{w'}(l_v^{w'})$ . To facilitate the downward pass of the algorithm, we will store which method (commitment

at  $v$  or not) generated the lower cost, as well as the descendants  $(l_v^{w'})$  when the second method generates the lower cost.

At internal nodes  $v$  where player  $i \neq 1$  takes an action, for any  $l \in S_w$  (for some child  $w$  of  $v$ ), 1 can only incentivize player  $i$  to prefer  $l$  by making it have a (weakly) higher  $u_i$  than the outcome resulting from every other action it can take. Again,  $l$  must be selected in the child  $w$  it descends from, and a cost of  $c_w(l)$  will be incurred for that. In the interest of minimizing costs, we will select the cheapest option from each  $w' \neq w$ . That is, for each  $w'$ , we will find the cheapest descendant that will not be strictly preferred to  $l$ ,  $l_v^{w'} = \arg \min_{(l' \in S_{w'} \text{ and } u_i(l') \leq u_i(l))} c_{w'}(l')$ , and its cost is  $c_{w'}(l_v^{w'})$ . Again,  $l_v^{w'}$  may not be well defined, because it may be that for all  $l' \in S_{w'}$ ,  $u_i(l') > u_i(l)$ , so that min is taken over an empty set. This corresponds to the case where it is impossible to make player  $i$  prefer  $l$  over  $w'$ ; in this case we exclude  $l$  from  $S_v$ . If  $l$  is not excluded in this way (for any  $w'$ ), we let  $l \in S_v$  and set  $c_v(l) = c_w(l) + \sum_{w' \neq w} c_{w'}(l_v^{w'})$ . Again, to facilitate the downward pass of the algorithm, we will store which descendants  $(l_v^{w'})$  are used.

This allows us to use dynamic programming to compute  $S_r$ ; we can then choose  $l^* \in \arg \max_{l \in S_r} \{u_1(l) - c_r(l)\}$ .

### **Commitment determination (downward pass)**

We need a procedure  $\text{strategy}(v, l)$  that, given a node  $v$  and one of its descendant leaves  $l \in S_v$ , specifies how to commit in the subtree rooted at  $v$  so that the outcome  $l$  results. Let  $w$  be the child of  $v$  that is an ancestor of  $l$ . If player 1 acts at  $v$  and the first method was used to generate  $c_v(l)$  (if  $c_v^1(l) \leq c_v^2(l)$ ), then we simply commit to going to  $w$  and recursively call  $\text{strategy}(w, l)$ . In this case, we do not commit to anything in the subtrees of the other children of  $v$ . If player 1 acts at  $v$  and the second method was used to generate  $c_v(l)$  (if  $c_v^1(l) > c_v^2(l)$ ), then we recursively call  $\text{strategy}(w, l)$ , and, for all  $w' \neq w$ , we call  $\text{strategy}(w', l_v^{w'})$ . Similarly, at a node where player  $i \neq 1$  acts, we recursively call  $\text{strategy}(w, l)$ , and, for all  $w' \neq w$ , we call  $\text{strategy}(w', l_v^{w'})$ .

## Runtime analysis

We recall that there are  $m$  leaves and  $n$  internal nodes. In the dynamic programming step, at each node  $v$ , for every child  $w$  of  $v$ , for every  $l \in S_w$ , we go through the elements of  $S_{w'}$  for each other child  $w' \neq w$  of  $v$ . As we do a constant amount of work each time and there are at most  $O(m)$  elements in each level of the tree, we can bound the runtime per node to  $O(m^2)$ , and hence for the dynamic programming pass as a whole to  $O(nm^2)$ . The commitment determination phase merely requires lookups.  $\square$

### 3.1.3 Case 3. No chance, pure strategies, two players, no costs or restrictions, perfect information

So far, we have focused on commitment to pure strategies in perfect-information games of no chance, represented in tree form. The following result extends this to DAG games, though here we must restrict to the two-player case without costs and restrictions (our later hardness results show that we cannot hope for more, unless  $P=NP$ ).

**Theorem 8.** *In two-player perfect-information games in DAG form with no chance nodes and no costs/restrictions on commitment, the optimal pure strategy to commit to can be found in time  $O(mn(m+n))$ .*

*Proof.* Let  $S$  be the set of all possible outcomes of the game (in a tree game, the outcomes would be the leaves of the tree; in a DAG, they are the sinks). We sort  $S$  by player 1's utility. Next, for each outcome  $l \in S$  (in order from highest to lowest  $u_1$ ), we test if it is possible to achieve this outcome; when the test turns out positive, that outcome will correspond to the solution.

The test for  $l$  is a bottom-up dynamic program. The intuition is as follows. For every node  $v$ , we check whether there exists a strategy to commit to for the subgame rooted at  $v$  that results in  $l$ . If not, the goal at this node becomes to minimize player 2's utility at it (so as to make this node as unattractive as possible higher up in the DAG). We let  $b(v) = 1$  if  $l$  is achievable from it, and  $b(v) = 0$  if not. In the latter case, we set  $m(v)$  to be the minimum

number  $m$  such that we can guarantee that player 2 gets utility at most  $m$  from  $v$ . For  $l$ , we set  $b(l) = 1$ ; for every outcome  $l' \neq l$ , we set  $b(l') = 0$  and  $m(l') = u_2(l')$ .

If 1 acts at  $v$ , then  $b(v) = 1$  if and only if for at least one child  $w$  of  $v$ ,  $b(w) = 1$ . (We note that there may be multiple such children in a DAG.) Player 1 can achieve  $l$  from  $v$  by committing to going to  $w$ . If not, it follows that for every child  $w$  of  $v$ ,  $b(w) = 0$ , and hence we have computed  $m(w)$  for each of them. We get  $m(v) = \min_{w \text{ child of } v} m(w)$ , which can be achieved by committing to going to some child  $w^* \in \arg \min_{w \text{ child of } v} m(w)$ .

If 2 acts at  $v$ , then  $b(v) = 1$  if and only if (1) there exists a child  $w$  of  $v$  such that  $b(w) = 1$  and (2) for all children  $w'$  where  $b(w') = 0$ , we have  $m(w') \leq u_2(l)$ . Otherwise,  $b(v) = 0$  and  $m(v) = \max_{w \text{ child of } v} m(w)$ .

Finally, if  $b(r) = 1$  for the root node  $r$ , then the algorithm terminates with the commitment strategy obtained with this dynamic program. If not, we move on to the next outcome (and since we consider all outcomes, eventually we must get  $b(r) = 1$  on a pass).

### **Runtime analysis**

The sorting at the start takes  $O(m \log(m))$  time. In each of the up to  $m$  dynamic programming iterations (one for each  $l$ ), at each of the  $n$  internal nodes  $v$ , we must take a minimum over all its children, of which there could be up to  $m+n$ . Thus the total runtime is  $O(mn(m+n))$ . □

#### *3.1.4 Case 4. No chance, trees, two-player, no costs or restrictions, perfect information*

So far, we have focused on commitment to pure strategies in perfect-information games of no chance. We now give a positive result for commitment to mixed behavioral strategies, although this requires that there be only two players and that the game be represented in tree form. (Again, our later hardness results show that we cannot hope for more, unless  $P=NP$ .) Our next result also assumes that there are no costs or restrictions; we will generalize to allow for them in the result following it.

**Theorem 9.** *In two-player perfect-information games in tree form with no chance nodes and no costs/restrictions on commitment, the optimal mixed strategy to commit to can be found in time  $O(nm^2)$ .*

*Proof.* The algorithm contains two main steps. First, we will determine through a bottom-up dynamic program, for each node  $v$ , a set  $S_v$ , which is the set of all mixtures over  $v$ 's descendant leaf nodes that can be achieved by an appropriate commitment by player 1. To aid in visualizing  $S_v$ , we picture each leaf node  $l$  as a point  $p$  in two-dimensional space, where the  $x$ -dimension is player 2's utility and the  $y$ -dimension is player 1's utility. Since the goal is a commitment strategy that maximizes utility for player 1, we need only maintain the upper envelope of  $S_v$ , which we can represent as a set of line segments, each of which can be stored compactly as pairs of two points  $(p, p')$ . Let  $S_v^1$  be the set of line segments that we maintain. (In fact, for computational simplicity, we will maintain more than just the line segments corresponding to the upper envelope, but the upper envelope of the line segments in  $S_v^1$  will correspond to the upper envelope of  $S_v$ .) Finally, for computational reasons, we maintain a set  $S_v^2$ , which is always a subset of the endpoints of the line segments in  $S_v^1$  (but excludes some that we will not need).

**Dynamic programming step (upward pass)**

For each leaf node  $l$ ,  $S_l^2 = \{l\}$  and  $S_l^1 = \{(l, l)\}$ . At a node  $v$  where player 1 acts, for any child  $w$  of  $v$ , any point in  $S_w$  is also in  $S_v$ . Moreover, if we take one point from  $S_w$  for every child  $w$ , every mixture over these points is also in  $S_v$ . However, because there are only two players, it will never be helpful to mix over points from more than two children (for any point obtained that way, there will be a point in  $S_v$  that is obtained by mixing over points from only two children that has the same utility for player 2 and at least the same utility for player 1). This allows us to compute the set of endpoints:  $S_v^2 = \bigcup_{w \text{ child of } v} S_w^2$ . The line segments in  $S_v^1$  will come from two different sources. First, some line segments will come directly from  $S_w^1$  for some child  $w$ , via a pure commitment to  $w$ . The second source

is by generating segments by mixing between any two children  $w$  and  $w' \neq w$ ; in this case, we can restrict attention to mixing over one end point from  $S_w^2$  and one from  $S_{w'}^2$  (points obtained by mixing over other points will be dominated). We can define the set resulting from this second source as  $\hat{S}_v^1 = \bigcup_{p \in S_w^2, p' \in S_{w'}^2 \text{ s.t. } w <_{w'} \{p, p'\}} \{(p, p')\}$  (we define an arbitrary order  $<$  over the child nodes to avoid duplication). Thus, we get that  $S_v^1 = (\bigcup_{w \text{ child of } v} S_w^1) \cup \hat{S}_v^1$ .

For a node  $v$  at which player 2 acts, it can happen that, for at least one child  $w$  of  $v$ , there are values in  $S_w$  that it is impossible to incentivize player 2 to play. Let  $m_{w'} \in \arg \min_{p \in S_{w'}^2} u_2(p)$ ; this point corresponds to the most we can “punish” player 2 for going to  $w'$ . Let  $i(w) = \max_{w' \neq w} u_2(m_{w'})$ ; this value corresponds to the smallest utility that player 2 will accept when going to  $w$ .

Based on  $i(w)$ , we will shrink  $S_w^1$  into a set  $\hat{S}_w^1$ , so that  $S_v^1 = \bigcup_{w \text{ child of } v} \hat{S}_w^1$ . For every  $(p, p') \in S_w^1$ , we do the following:

- If  $u_2(p) \geq i(w)$  and  $u_2(p') \geq i(w)$ , add  $(p, p')$  to  $\hat{S}_w^1$ .
- If  $u_2(p) < i(w)$  and  $u_2(p') < i(w)$ , do not add  $(p, p')$  to  $\hat{S}_w^1$ .
- If  $u_2(p) < i(w)$  and  $u_2(p') \geq i(w)$ , find the point  $p''$  where  $u_2(p'') = i(w)$  and  $p'' = (\alpha p + (1 - \alpha)p')$  for some  $\alpha \in [0, 1]$ . Then, add  $(p'', p')$  to  $\hat{S}_w^1$  and add  $p''$  a set of new potential endpoints  $\hat{S}_w^2$ . The case where  $u_2(p) \geq i(w)$  and  $u_2(p') < i(w)$  is similar.

To calculate  $S_v^2$ , we include all points  $p$  that are in  $S_w^2$  for some child  $w$  and for which  $u_2(p) \geq i(w)$ . Additionally, for each child  $w$ , if  $\hat{S}_w^1$  is nonempty, we add one point  $p$  to  $S_v^2$ , namely the point in  $\arg \max_{p \in \hat{S}_w^1} u_1(p)$ .

Finally, at the root node  $r$ , we calculate the solution

$$p^* \in \arg \max_{p \in S_r^2} u_1(p).$$

**Commitment determination (downward pass)**

We need a procedure  $\text{strategy}(v, p'')$  that, given a node  $v$  and a point  $p''$  that lies on one of the lines  $(p, p') \in S_v^1$ , specifies how to commit in the subtree rooted at  $v$  so that the outcome  $p$  results. If player 1 acts at  $v$ , we calculate  $\alpha$  and  $(p, p')$  where  $p'' = \alpha p + (1 - \alpha)p'$ ,  $\alpha \in [0, 1]$ . Then, we find  $w$  and  $w'$  where  $p \in S_w^2$  and  $p' \in S_{w'}^2$ . If  $w = w'$ , simply commit to going to  $w$  and recursively call  $\text{strategy}(w, p'')$ . If  $w \neq w'$ , then at  $v$  commit to the mixture  $\alpha w$  and  $(1 - \alpha)w'$ , and recursively call  $\text{strategy}(w, p)$  and  $\text{strategy}(w', p')$ . (It does not matter how 1 commits at other children of  $v$  since they will not be reached.) For a node  $v$  where player 2 acts, find the descendant  $w$  that contains  $p''$  (more precisely, find  $w$  such that  $(p, p') \in S_w^1$  where  $p'' = \alpha p + (1 - \alpha)p'$  for some  $\alpha \in [0, 1]$ ). Then, recursively call  $\text{strategy}(w, p'')$ , and, for all other children  $w' \neq w$ , call  $\text{strategy}(w', m_w)$ .

### Runtime analysis

If we consider all the  $S_v^2$  on a single level of the tree, we can bound the sum of the sizes of these sets by  $m$ , because there are only two types of points in these sets. First, any leaf node can appear in them (but can appear only once at each level of the tree). Second, at any player 2 node at or below this level, a new point can be created, but if so, another point from below this node will be removed.

Next, let us consider the potential sum of the sizes of all the  $S_v^1$  on a single level. New line segments can only be generated at internal nodes  $v$  from some  $p \in S_w^2$  and some  $p' \in S_{w'}^2$ , where  $w$  and  $w' \neq w$  are children of  $v$ , and thus at the same level. If the only end points ever generated were leaf nodes (which is not true in general), then there can be at most  $\binom{m}{2}$  line segments at a level (since there will be no duplicates). More generally, if a new end point is created at a player 2 node, it replaces another end point; therefore, every end point corresponds to a leaf that it replaced (or that was replaced by the end point it replaced, etc.). Therefore the  $\binom{m}{2}$  bound on the number of line segments at a level holds generally.

At a node  $v$  where player 1 acts, taking an existing line segment from one of its children or generating a new one from two of its children is a constant-time operation per line

segment. Hence, this step takes  $O(m^2)$  time per node. At a node  $v$  where player 2 acts, it also takes  $O(m^2)$  time to evaluate all the line segments from all of its children to see which have to be altered and which can be included without change. Thus, the upward pass requires  $O(nm^2)$  time. The downward pass merely has to find, at each node  $v$ , which line segment(s) intersects with the desired point, which can be done in  $O(m^2)$  time, as there are at most  $O(m^2)$  line segments in its children's feasible sets. Again, because we are doing this for at most  $O(n)$  nodes, this requires at most  $O(nm^2)$  time in total.  $\square$

### 3.2 Negative results

In this section we give NP-hardness results for the following problems:

Case 1. It is NP-hard to solve for the optimal strategy to commit to when chance nodes are allowed in two-player games in tree form even in perfect information games with no costs/restrictions, regardless of whether commitment to mixed strategies is allowed (Theorem 10).

Case 2. It is NP-hard to solve for the optimal mixed strategy to commit to in three-player games, even in perfect-information games in tree form with no chance nodes and no costs/restrictions (Theorem 11).

Case 3. It is NP-hard to solve for the optimal strategy to commit to in two-player games of imperfect-information in tree form, even in games with no chance nodes and no costs/restrictions, regardless of whether commitment to mixed strategies is allowed (Theorem 12).

Case 4. It is NP-hard to solve for the optimal strategy to commit to in two-player games in DAG form even in perfect information games with no chance nodes and no costs/restrictions, when commitment to mixed strategies is allowed (Theorem 13).

Case 5. It is NP-hard to solve for the optimal strategy to commit to in three-player games in DAG form, even in perfect-information games with no chance nodes and no costs/restrictions, regardless of whether commitment to mixed strategies is allowed. (The-

orem 14).

Case 6. It is NP-hard to solve for the optimal strategy to commit to in three-player games in DAG form with commitment restrictions, even in perfect-information games with no chance nodes, regardless of whether commitment to mixed strategies is allowed. (Theorem 15).

### 3.2.1 Case 1. Chance nodes

Our first hardness result shows that when chance nodes are involved, even the simplest of problems become NP-hard.

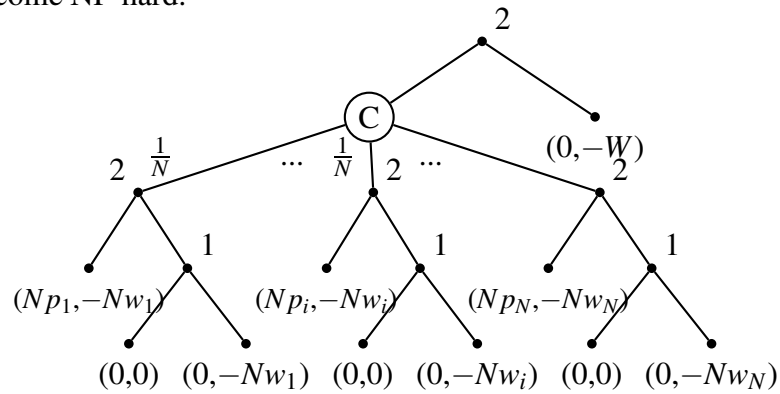


FIGURE 3.3: Two-player game with chance nodes used in the hardness reduction of Theorem 10.

**Theorem 10.** *It is NP-hard to solve for the optimal strategy to commit to in a game with chance nodes, even if the game has only two players, it is in tree form, it is a game of perfect information with no costs/restrictions, and regardless of whether commitment to mixed strategies is allowed.*

*Proof.* In the KNAPSACK problem, we are given a set of  $N$  items, and for each of them, a value  $p_i$  and a weight  $w_i$ ; additionally, we are given a weight limit  $W$ . We are asked to find a subset of the items with total weight at most  $W$  that maximizes the sum of the  $p_i$  in the subset. We reduce an arbitrary KNAPSACK instance to an extensive-form game, in such a way that the maximal utility obtainable by the leader with commitment (whether

pure or mixed) is equal to the optimal solution value in the KNAPSACK instance. The extensive-form game is illustrated in Figure 3.3, and defined formally below.

### **Top of the tree**

The first move is by player 2, who chooses between an outcome of  $(0, -W)$ , and a chance node which randomizes uniformly over the item subtrees, defined next.

### **Item subtrees**

Each item subtree  $t^i$  has two levels. At the top level, there is a node where player 2 acts. It has two children: one is a leaf node with an outcome of  $(Np_i, -Nw_i)$ , the other is a node where player 1 acts. The latter node also has two children: leaf nodes  $(0, 0)$  and  $(0, -Nw_i)$ .

### **Proof of equivalence to KNAPSACK instance**

If, at the lowest internal node of an item subtree, player 1 commits to playing 100% Right, then player 2, breaking ties in 1's favor, will move Left in this subtree, resulting in payoffs  $(Np_i, -Nw_i)$  if this subtree is reached. Otherwise, player 2 will move Right, and player 1 will get 0 (and player 2 at most 0). Because player 1 wants player 2 to move to the chance node at the top, there is no benefit to player 1 in moving Right with probability strictly between 0 and 100%, since this will only make the chance node less desirable to player 2 without benefiting player 1, so we can assume without loss of optimality that player 1 commits to a pure strategy.

Let  $S$  be the set of indices of subtrees where player 1 commits to playing Right. Then, player 2's expected utility for the chance node is  $(1/N) \sum_{i \in S} -Nw_i = -\sum_{i \in S} w_i$ . Player 2 will choose to move to the chance node if and only if  $\sum_{i \in S} w_i \leq W$ . Given this, player 1's expected utility is  $(1/N) \sum_{i \in S} Np_i = \sum_{i \in S} p_i$ . Hence, finding player 1's optimal strategy to commit to is equivalent to solving the KNAPSACK instance.  $\square$

*Aside: Chance Node Gadget*

Before we get to the next reduction, let us introduce a gadget that we will use in the next two reductions. This gadget allows us to create a pseudo-chance node in games of no chance, when commitment to mixed behavioral strategies is allowed. The construction of a pseudo-chance node with  $N$  equally likely outcomes requires  $2(N - 1)$  internal nodes and is as pictured in Figure 3.4. The outcomes of the chance node correspond to the leaves immediately following an action by player 1.

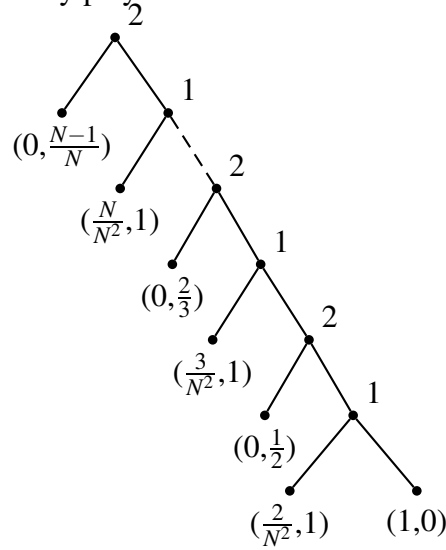


FIGURE 3.4: Pseudo-chance node gadget used in Theorem 11 and Theorem 13.

To illustrate how the construction works, let us solve for the optimal mixed behavioral strategy for player 1 to commit to.

**Proposition 1.** *The optimal mixed behavioral strategy for player 1 to commit to in the game in Figure 3.4 is the following: At the  $i$ th node for player 1 from the bottom, commit to the mixture  $(1/(i + 1), i/(i + 1))$ . Given this, player 2 will always move right, and each leaf node directly connected to a player 1 node is reached with probability  $1/N$ . Moreover, if the utilities for player 1 are perturbed by a sufficiently small amount, the optimal mixed behavioral strategy for player 1 remains exactly the same.*

*Proof.* In an optimal solution, player 1 will incentivize player 2 to move Right everywhere, because player 1 gets 0 when player 2 moves Left; hence, given a solution where player 2 moves Left somewhere, that solution can be (weakly) improved by changing the commitment further down the tree so that player 2 will in fact move Right at that node.

Let us start by considering player 1's optimal commitment at the bottom internal node of the tree. Player 1 needs to commit to placing at least probability  $1/2$  on Left for player 2 to be willing to move Right at the node above this one. However, Right gives player 1 a larger utility. Hence, if this subtree (starting at player 2's node) were the entire game, player 1 will want to commit to the mixture of  $1/2$  Left,  $1/2$  Right at this bottom internal node.

Next, let us consider a larger subgame, including the next two levels up (one level for each player). To incentivize player 2 to move Right at the root of this subgame player 1 needs to commit in such a way that player 2 receives an expected utility of at least  $\frac{2}{3}$  in the right subtree. Player 1 can increase player 2's expected utility for moving Right in two ways, by increasing the chance of realizing the outcome  $(\frac{3}{N^2}, 1)$  or by increasing the chance of realizing the outcome  $(\frac{2}{N^2}, 1)$ . Since these outcomes only differ in player 1's utility, player 1 receives a higher expected utility by incentivizing player 2 via  $(\frac{3}{N^2}, 1)$  than via  $(\frac{2}{N^2}, 1)$ . Of course, we still have the constraint that, at the bottom level, player 1 needs to place probability at least  $\frac{1}{2}$  on Left (the outcome  $(\frac{2}{N^2}, 1)$ ). So, in the optimal solution for this subgame, player 1 places probability exactly  $\frac{1}{2}$  on Left at the bottom level, and then at the next level up for player 1, she places probability  $\frac{1}{3}$  on Left, which makes player 2 indifferent between moving Left and Right at the root of this subgame.

This logic can be applied inductively to each pair of levels up the tree, leading to an optimal strategy where player 1's commitment at her  $i$ th node is to put probability  $1/(i+1)$  on Left, so that player 2 is always incentivized to go Right. Hence, the overall probability that the game ends by player 1 moving Left at her  $i$ th node from the bottom (and not

before) is  $\frac{1}{i+1} \prod_{j=i+1}^{N-1} j/(j+1) = 1/N$ ; and the probability that the game ends without any moves to the Left (resulting in the outcome  $(1,0)$ ) is  $\prod_{j=1}^{N-1} j/(j+1) = 1/N$ . Because in this proof, we only used ordinal comparisons between player 1's utilities at the leaves, the proof is robust to small perturbations of player's 1's utilities that do not affect the ordinal comparisons.  $\square$

Thus, this construction allows us to generate a uniform distribution. This may at first appear not nearly as useful as being able to use a true chance node, because we can only get a uniform distribution over specially chosen leaves, rather than over any collection of subtrees. Nevertheless, in the proofs below, we will be able to modify some of these leaves into more interesting subtrees without affecting the strategic result of a uniform distribution.

### 3.2.2 Case 2. Three-players, mixed strategy commitment

We now use the pseudo-chance node gadget to prove that optimal commitment to a mixed behavioral strategy is NP-hard in three-player games.

**Theorem 11.** *It is NP-hard to solve for the optimal mixed strategy to commit to in three-player games, even in perfect-information games in tree form with no chance nodes and no costs/restrictions.*

*Proof.* In the KNAPSACK problem, we are given a set of  $N$  items, and for each of them, an integer value  $p_i$  and an integer weight  $w_i$ ; additionally, we are given an integer weight limit  $W$ . (Without loss of generality,  $w_i \leq W$  for all  $i$ .) We are asked to find a subset of the items with total weight at most  $W$  that maximizes the sum of the  $p_i$  in the subset. Let  $p^* = \max_i p_i$ . We reduce an arbitrary KNAPSACK instance to a three-player extensive-form game, in such a way that the maximum utility obtainable by the leader with mixed

strategy commitment corresponds to the maximum value obtainable in the KNAPSACK problem. The extensive-form game is illustrated in Figure 3.5, and defined formally below.

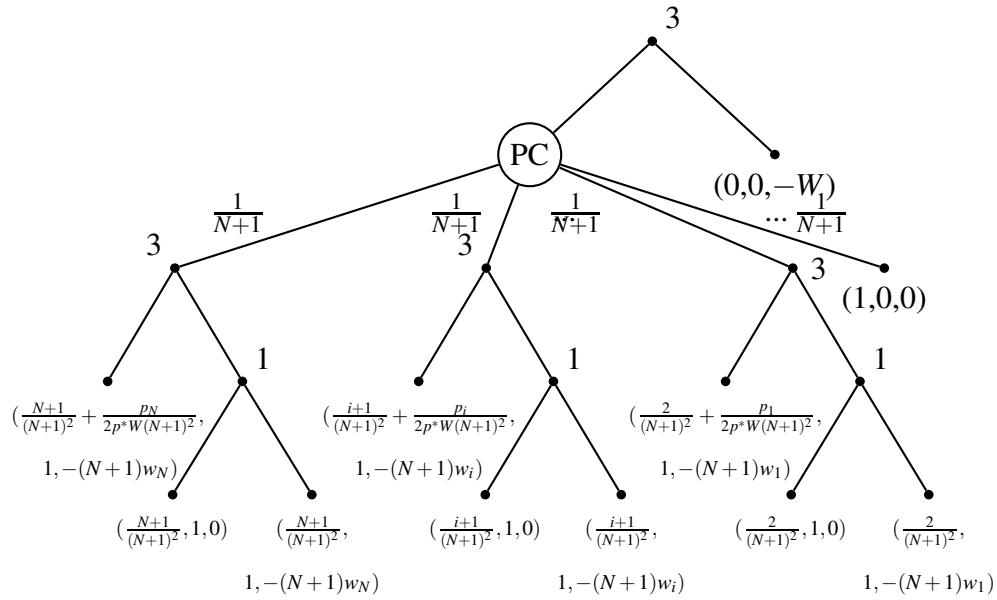


FIGURE 3.5: Three-player tree with pseudo-chance node used in the hardness reduction of Theorem 11.

### Top of the tree

The first move is by player 3, who can choose either an outcome of  $(0, 0, -W)$ , or to move to the pseudo-chance node. The pseudo-chance node will (in the optimal mixed commitment solution) randomize uniformly over  $N$  item subtrees and an additional outcome of  $(1, 0, 0)$ . (In the representation of the tree in Figure 3.5, the children of the pseudo-chance node PC correspond to all the leaves of the tree in Figure 3.4 where player 1 moved last. The leaves where player 2 moved last still need to be part of the game, but we do not draw them in Figure 3.5 since they will not be reached in the optimal solution—they are “hidden” as part of the pseudo-chance node construction. This is the reason that player 2 does not occur at all in Figure 3.5—he is “hidden in the pseudo-chance node.” Also note that the use of the pseudo-chance node places restrictions on the utilities below it: player 2’s utilities must be identical to those in Figure 3.4 (that is, they must all equal 1), and player

1's utilities must be very close to those in Figure 3.4. Note that the  $N$  in Figure 3.4 equals  $N + 1$  in Figure 3.5.)

### Item subtrees

We define a subtree  $t^{x_i}$  for each integer. This subtree consists of two levels. At the top node of the subtree, player 3 chooses between two children: one is a leaf node with an outcome of  $(\frac{i+1}{(N+1)^2} + \frac{p_i}{2p^*W(N+1)^2}, 1, -(N+1)w_i)$ , the other is a node where player 1 acts. This latter node also has two children: leaf nodes  $(\frac{i+1}{(N+1)^2}, 1, 0)$  and  $(\frac{i+1}{(N+1)^2}, 1, -(N+1)w_i)$ .

### Proof of equivalence to KNAPSACK instance

If at the lowest internal node of an item subtree, player 1 commits to playing 100% Right, then player 3 will move Left in this subtree, resulting in payoffs

$(\frac{i+1}{(N+1)^2} + \frac{p_i}{2p^*W(N+1)^2}, 1, -N+1w_i)$ . Otherwise, player 3 will move Right, and player 1 will get  $\frac{i+1}{(N+1)^2}$  (and player 3 at most 0). Because player 1 wants player 3 to move to the pseudo-chance node at the top, there is no benefit to player 1 in moving Right with probability strictly between 0 and 100%, since this will only make the pseudo-chance node less desirable to player 3, without benefiting player 1, so we can assume without loss of optimality that player 1 commits to a pure strategy in each item subtree.

Let us assume, for now, that in the optimal solution, the distribution over the pseudo-chance node's children is uniform. Let  $S$  be the set of indices of subtrees where player 1 commits to playing Right. Then, player 3's expected utility for the chance node is  $(1/(N+1))\sum_{i \in S} -(N+1)w_i = -\sum_{i \in S} w_i$ . Player 3 will choose to move to the chance node if and only if  $\sum_{i \in S} w_i \leq W$ . Given this, player 1's expected additional utility (relative to the case where  $S = \emptyset$ ) is  $(1/(N+1))\sum_{i \in S} \frac{p_i}{2p^*W(N+1)^2} = \frac{1}{(2p^*W(N+1)^3)}\sum_{i \in S} p_i$ . This quantity is maximized when  $\sum_{i \in S} p_i$  is maximized (nothing else depends on  $S$ ), so finding player 1's optimal mixed strategy to commit to is equivalent to solving the KNAPSACK instance—under the assumption that the distribution over the pseudo-chance node's chil-

dren is uniform.

Now, let us revisit that assumption—can player 1 obtain even higher utility by committing inside the pseudo-chance node in a way that results in the pseudo-chance node not mixing uniformly?

First, suppose that the choice of player 1’s commitment in the item subtrees is fixed to a feasible solution  $S$ , that is,  $\sum_{i \in S} w_i \leq W$ . Then, by the analysis in the proof of Proposition 1, player 1 is best off committing in the pseudochance node in a way that results in a uniform distribution, because player 1’s utilities are only slightly perturbed relative to Figure 3.4.

The more difficult case is when player 1’s commitment in the item subtrees is fixed to an *infeasible* solution  $S$ , with  $\sum_{i \in S} w_i > W$ . While this is clearly not beneficial for player 1 if the pseudochance node results in a uniform distribution (because player 3 will move Right at the beginning), it may be possible for player 1 to commit to a different strategy in the pseudo-chance node so that player 3 will in fact move Left at the beginning of the game. We will show that player 1 cannot benefit from such an approach, in the sense that in this case player 1 would be better off reducing  $S$  to the point where it is feasible, and committing in a way such that the pseudo-chance node results in a uniform distribution. Roughly, the idea is the following. Suppose that  $\sum_{i \in S} w_i > W$ . By integrality, we know that  $\sum_{i \in S} w_i \geq W + 1$ , that is, the sum of the included items’ weights is bounded away from  $W$ . Hence, in order to make player 3 play Left at the root, player 1’s strategy at the pseudo-chance node must be significantly different, so that the resulting distribution over the children of the pseudo-chance node must be significantly different from the uniform distribution. We then show that the cost of this difference to player 1 is larger than what she could ever obtain from including more items.

Specifically, in any distribution over the pseudo-chance node’s children that makes player 3 play Left when  $\sum_{i \in S} w_i > W$ , at least  $1/W$  of the probability mass must have shifted relative to the uniform distribution over the pseudo-chance node’s children. This is because player 3’s expected utility for going Left must be increased by at least 1 relative

to the uniform distribution, but for each unit of mass that is shifted, the expected utility of player 3 can change by at most  $\max_i w_i \leq W$ .

Moreover, player 1's commitment in the pseudo-chance node must still incentivize player 2 to always move Right, because player 1 gets 0 at any leaf that follows a Left move by player 2. It is not difficult to show that in order for this to be true, it must be the case that the first  $i$  children of the pseudo-chance node receive probability at most  $i/N$  (and they receive exactly this probability in the uniform distribution). For any alternate probability distribution where this condition holds, there exists a way to transform the uniform distribution to this alternate distribution while only shifting probability mass from smaller indices to larger indices. When we shift mass in this way, all of the mass that is shifted is shifted to an index that is at least 1 larger. It follows that if  $1/W$  mass is shifted in total from the uniform distribution, then it must be the case that the expected index of the child chosen from the pseudo-chance node increases by at least  $1/W$ . As a result, the expectation of the term  $(i+1)/(N+1)^2$  in player 1's payoff decreases by at least  $1/(W(N+1)^2)$ . (Note that  $N+2-i$  is the index of the child.)

On the other hand, the expectation of the term  $x_i \cdot p_i / (2p^*W(N+1)^2)$  in player 1's expected utility—where  $x_i \in \{0, 1\}$  indicates whether item  $i$  has been included in  $S$ , so that this term corresponds to her utility from including items—can increase by at most  $\max_i p_i / (2p^*W(N+1)^2) = p^* / (2p^*W(N+1)^2) = 1/(2W(N+1)^2)$ , which is less than  $1/(W(N+1)^2)$ . Hence it follows that it is never helpful to player 1 to change the distribution from the pseudo-chance node in order to include items beyond the limit  $W$ . It follows that the problem of finding the optimal mixed strategy to commit to in the game in Figure 3.5 is equivalent to the KNAPSACK instance.  $\square$

### 3.2.3 Case 3. Two players, imperfect information, pure or mixed strategy commitment

Next, we will show that the addition of imperfect information is enough to make the problem NP-hard.

**Theorem 12.** *It is NP-hard to solve for the optimal strategy to commit to in two-player games of imperfect information in tree form, even in games with no chance nodes and no costs/restrictions, regardless of whether commitment to mixed strategies is allowed. This holds even if only player 1 has non-singleton information sets.*

*Proof.* We will first consider the case of mixed-strategy commitment. We reduce an arbitrary instance of 3SAT to an extensive-form game such that the leader can obtain utility 1 if the 3SAT instance is satisfiable and 0 otherwise. The 3SAT instance consists of  $N$  variables  $x_1, \dots, x_N$  and  $M$  clauses,  $C_1, \dots, C_M$ . This game is illustrated in Figure 3.6 and the details of its construction follow.

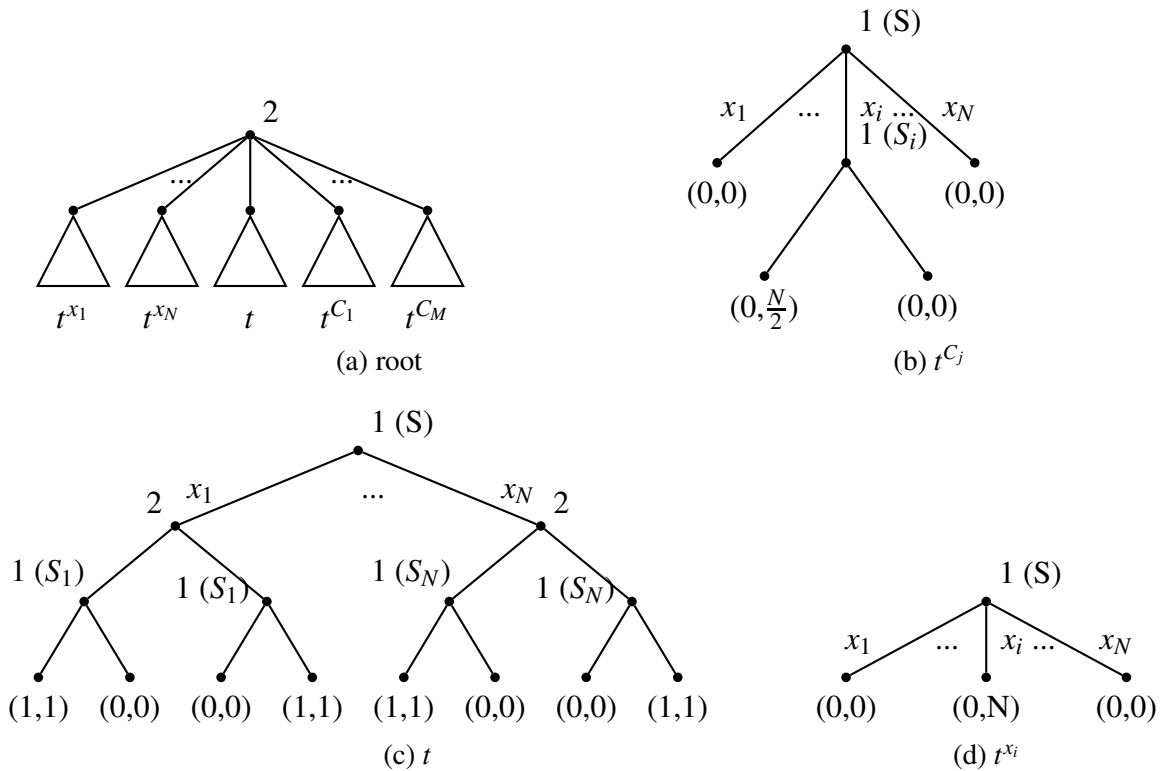


FIGURE 3.6: Two-player game of imperfect information used in the hardness reduction of Theorem 12. The notation (X) next to a node indicates that the node is part of the information set X. In  $t^{C_j}$ ,  $+x_i$  appears in  $C_j$ .

### The top of the tree and the information sets

Connected to the root (at which player 2 acts), for each variable  $x_i$ , the game contains a subtree  $t^{x_i}$ ; for each clause  $C_i$ , a subtree  $t^{C_j}$ ; and finally, a “target” subtree  $t$ . Player 1’s objective is to incentivize player 2 to choose  $t$ , as this is the only subtree where she can get positive utility. The roots of all of these subtrees belong to player 1, and they are all in the same information set  $S$ . Thus, they have the same possible actions for player 1, specifically, one action for each variable  $x_i$ . Furthermore, we have one information set  $S_i$  for each variable  $x_i$ ; the commitment decision here will correspond to the choice of a literal ( $+x_i$  or  $-x_i$ ).

### **Target subtree**

The root of the  $t$  subtree has a child for each variable  $x_i$ ; each of these corresponds to a simple coordination game, where 1 and 2 both receive 1 if and only if they choose the same action. There is one twist: player 1’s information set  $S_i$  in the coordination game for  $x_i$  also includes some nodes from the clause subtrees.

### **Clause subtrees**

The root of each  $t^{C_j}$  subtree also has a child for each variable  $x_i$ . If  $x_i$  does not appear in  $C_j$  at all, then this child is a leaf node with utilities  $(0,0)$ . If  $+x_i$  or  $-x_i$  appears in  $C_j$ , then the child is part of the information set  $S_i$ , and player 1 must move Left or Right. Moving Right corresponds to setting the variable to *true*, and results in utilities  $(0, \frac{N}{2})$  if  $-x_i$  appears in  $C_j$  and utilities  $(0,0)$  if  $+x_i$  appears in  $C_j$ . Moving Left corresponds to setting the variable to *false*, and results in utilities  $(0,0)$  if  $-x_i$  appears in  $C_j$  and utilities  $(0, \frac{N}{2})$  if  $+x_i$  appears in  $C_j$ .

### **Variable subtrees**

For each variable subtree  $t^{x_i}$ , the action corresponding to  $x_i$  leads to a leaf node with a payoff of  $(0,n)$ , and all other actions lead to a leaf node with a payoff of  $(0,0)$ .

### **Proof of equivalence to 3SAT instance (with mixed-strategy commitment)**

First, in order for player 2 to choose  $t$ , player 1 must commit to a mixed behavioral strategy that places equal  $(\frac{1}{N})$  probability on each of the choices in the information set

$S$ . Otherwise, some  $x_i$  receives more than  $\frac{1}{N}$  probability, and as a result player 2 would receive strictly more than 1 expected utility from  $t^{x_i}$ , whereas  $t$  can give player 2 at most 1. Given this, player 1 also must commit to a pure strategy in each of the information sets  $S_i$ . This is because otherwise, player 2 would get expected utility strictly less than 1 for  $t$  (because with some positive probability there will be a coordination failure), causing him again to prefer some  $t^{x_i}$ .

Thus, the only decision remaining for player 1 is whether to commit to Left or Right at each  $S_i$  (that is, whether to choose  $-x_i$  or  $+x_i$ , respectively). We now argue that player 2 will choose  $t$  rather than one of the clause subtrees, if and only if player 1's choices correspond to a satisfying assignment. If, in information set  $S_i$ , player 1 has committed to going Right ( $+x_i$ ), then all clause subtrees  $t^{C_j}$  with  $+x_i \in C_j$  will give player 2 a utility of at most 1. This is because in this case there will be at most two subtrees of the clause subtree where player 2 receives  $N/2$ , resulting in an expected utility of at most 1 for him for the clause subtree. The case where player 1 has committed to going Left ( $-x_i$ ) has an identical effect on the clause subtrees  $t^{C_j}$  with  $-x_i \in C_j$ . However, if all three literals of a clause  $C_j$  are unsatisfied, then player 2 will prefer this clause tree  $t^{C_j}$  to  $t$ , because his expected utility for  $t^{C_j}$  is  $\frac{3}{2} > 1$ .

Thus, the only way in which player 1 can obtain a utility of 1 is by committing to a uniform distribution in information set  $S$ , and a pure action at every information set  $S_i$ , where these actions correspond to a satisfying assignment for the 3SAT instance.

### **Commitment to pure actions only**

We now argue that player 1 can still obtain utility 1 with a satisfying assignment even if she can only commit to pure actions (but is free to not commit at some of the information sets). Commitment at information sets  $(S_1, \dots, S_N)$  is already restricted to pure actions in order for player 1 to obtain utility 1. However, player 1 is now unable to commit to a uniform distribution at information set  $S$ , and we have already seen that commitment to anything other than a uniform distribution here results in utility 0 for player 1. However,

if player 1 chooses to *not* commit in information set  $S$ , it turns out that player 1 playing a uniform distribution in information set  $S$  and player 2 playing choosing  $t$  is part of a Nash equilibrium, if and only if player 1 plays in a way that satisfies all clauses in the lower information sets. (In fact, player 1 does not need to commit in these information sets either, in the sense that there is an equilibrium in which she will play the satisfying assignment—of course, there are many other equilibria as well.) Thus, even in the pure-strategy case (where commitment is not required), player 1 can obtain a utility of 1 if and only if there is a satisfying assignment to the 3SAT problem.  $\square$

### 3.2.4 Case 4. Two-players, DAG, mixed strategy commitment

We again use the pseudo-chance node gadget from Subsection 3.2.1 to prove that optimal commitment to a mixed behavioral strategy is NP-hard in two-player games on a DAG.

**Theorem 13.** *It is NP-hard to solve for the optimal strategy to commit to in two-player games in DAG form even in perfect information games with no chance nodes and no costs/restrictions, when commitment to mixed strategies is allowed.*

*Proof.* We reduce an arbitrary instance of SAT to an extensive-form game with the properties in the theorem, so that the leader can obtain utility at least  $\frac{(M+1)(M+2)-2}{2(M+1)^3} + \frac{1}{M+1}$  if and only if the SAT instance is satisfiable. The SAT instance consists of  $N$  variables,  $x_1, \dots, x_N$ , and  $M$  clauses,  $C_1, \dots, C_M$ . The extensive-form game is illustrated in Figure 3.7 and the details of its construction are as follows:

#### Top of the game

The root is the pseudo-chance node, which randomizes over an outcome of  $(1, 0)$  and the clause subtrees  $t^{C_j}$ , which we define next.

#### Clause subtrees

For every clause  $C_j$ , there is a subtree  $t^{C_j}$ . In each such subtree, player 2 has a choice between: any literal subtree  $t^{+\{x_i \in C_j\}_k}$  where  $x_i$  is a positive literal and the  $k$ th literal of  $C_j$ ,

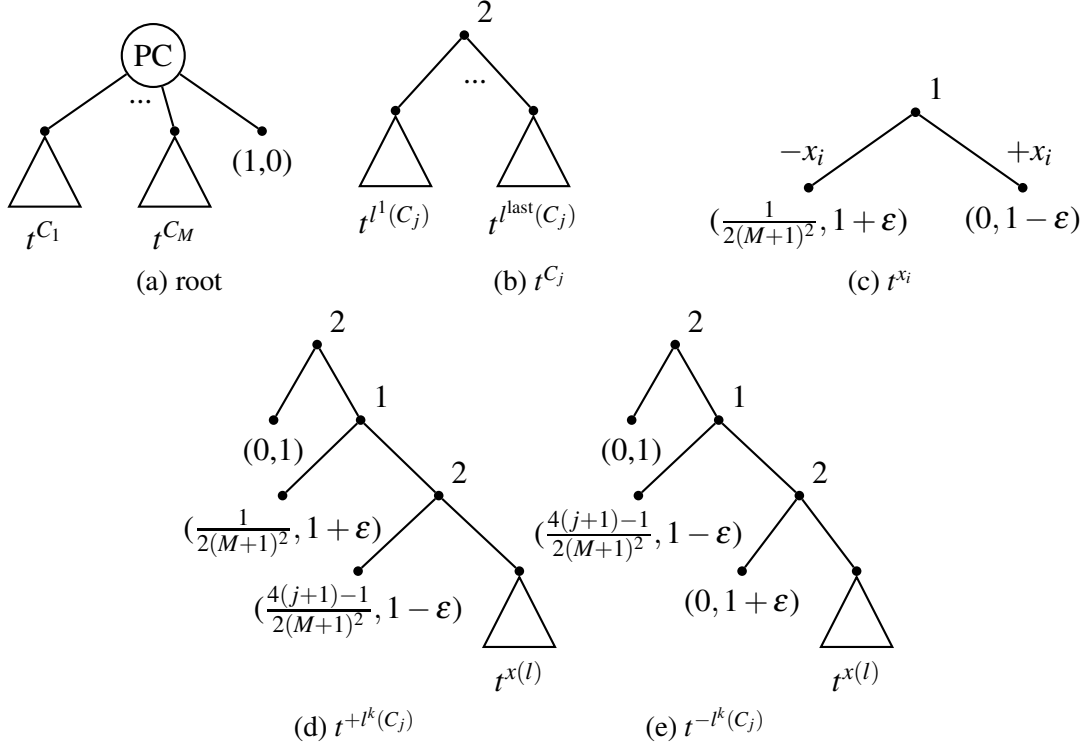


FIGURE 3.7: Two-player DAG game used in the hardness reduction of Theorem 13. Note that  $l^1(C_j)$  is the first literal in clause  $C_j$ ,  $l^{last}(C_j)$  is the last literal in clause  $C_j$ , and  $x(l)$  is the variable corresponding to literal  $l$ .

and any literal subtree  $t^{-\{x_i \in C_j\}_k}$  where  $x_i$  is a negative literal and the  $k$ th literal in  $C_j$ .

### Literal subtrees

For each occurrence of a positive literal  $+l^k(C_j)$  or a negative literal  $-l^k(C_j)$  in a given clause  $C_j$  we will have a distinct subtree ( $t^{+l^k(C_j)}$  or  $t^{-l^k(C_j)}$ ). In a literal subtree corresponding to a positive literal ( $t^{+l^k(C_j)}$ ), the first move is by player 2, who chooses between an outcome of  $(0, 1)$  and a choice by player 1. This choice by player 1 is between an outcome of  $(\frac{1}{2(M+1)^2}, 1 + \epsilon)$  and a choice by player 2. This final choice in this subtree is between an outcome of  $(\frac{4(j+1)-1}{2(M+1)^2}, 1 - \epsilon)$  and the variable subtree  $t^{x(l)}$  (described next). In a literal subtree corresponding to a negative literal ( $t^{-l^k(C_j)}$ ), the first move is by player 2, who chooses between an outcome of  $(0, 1)$  and a choice by player 1. This choice by

player 1 is between an outcome of  $(\frac{4(j+1)-1}{2(M+1)^2}, 1 - \varepsilon)$  and a choice by player 2. This final choice in this subtree is between an outcome of  $(0, 1 + \varepsilon)$  and the variable subtree  $t^{x(l)}$ .

### Variable subtrees

There is a single variable subtree  $t^{x_i}$  for each variable  $x_i$ . Note that this variable subtree can be reached from any (literal, clause) subtree where that literal corresponds to this variable—this is where the DAG aspect of the game is used. It will contain a single choice for player 1 between a pair of outcomes,  $(\frac{1}{2(M+1)^2}, 1 + \varepsilon)$  and  $(0, 1 - \varepsilon)$ .

### Proof of equivalence to SAT instance

In the variable subtree  $t^{x_i}$ , player 1 committing to choose the outcome  $(\frac{1}{2(M+1)^2}, 1 + \varepsilon)$  with probability 1 corresponds to setting variable  $x_i$  to false (so that  $\neg x_i$  is true), whereas committing to choose the outcome  $(0, 0)$  with probability 1 corresponds to setting variable  $x_i$  to true (so that  $\neg x_i$  is false). (We will argue that it is suboptimal to set the probability at any intermediate value.)

Consider first a subtree  $t^{-l^k(C_j)}$  corresponding to a negative literal. If  $x_i$  is set to false, then at the bottom of this subtree, player 2 has a choice between the outcomes  $(0, 1 + \varepsilon)$  and  $(\frac{1}{2(M+1)^2}, 1 + \varepsilon)$ , and will choose the latter (breaking ties in player 1's favor). On the other hand, if player 1 has committed to anything else in this variable subtree, player 2 will choose  $(0, 1 + \varepsilon)$ . In order to make player 2 move Right at the top of the negative literal subtree, at the next node, player 1 needs to put probability at least  $1/2$  on going Right (and will prefer to put no more than that). This results in an expected utility for player 1 for this subtree of  $\frac{(j+1)}{(M+1)^2}$  if  $x_i$  is set to false, and an expected utility of  $\frac{(j+1)}{(M+1)^2} - \frac{1}{4(M+1)^2}$  otherwise.

Now, let us consider a subtree  $t^{+l^k(C_j)}$  corresponding to a positive literal. If  $x_i$  is set to true, then at the bottom of this subtree, player 2 has a choice between the outcomes  $(0, 1 - \varepsilon)$  and  $(\frac{4(j+1)-1}{2(M+1)^2}, 1 - \varepsilon)$ , and will choose the latter (breaking ties in player 1's favor). On the other hand, if player 1 has committed to anything else in this variable

subtree, player 2 will choose the variable subtree, which results in a utility of at most  $\frac{1}{2(M+1)^2}$  for player 1. In this latter case, clearly player 1 can obtain an expected utility of at most  $\frac{1}{2(M+1)^2}$  from this positive literal subtree. In contrast, if  $x_i$  is set to true, then in order to make player 2 move Right at the top of the positive literal subtree, at the next node, player 1 needs to put probability at least  $1/2$  on going Left (and will prefer to put no more than that). This results in an expected utility for player 1 for this subtree of  $\frac{(j+1)}{(M+1)^2}$  if  $x_i$  is set to true.

Now let us consider a subtree  $t^{C_j}$  corresponding to a clause  $C_j$ . Given our previous analysis, all of the literal subtrees have a utility of 1 for player 2. Because player 2 breaks ties in player 1's favor, if any of the literal subtrees of a clause would give player 1 a utility of  $\frac{(j+1)}{(M+1)^2}$  (that is, if the clause is satisfied), then player 2 will choose such a subtree. Otherwise, player 1 necessarily obtains strictly less utility than that from this clause subtree. Hence, because the pseudochance node mixes over all the subtrees, if there exists a satisfying assignment for the SAT instance, player 1 can commit accordingly and obtain an expected utility of exactly  $\frac{1}{(M+1)}(1 + \sum_{j=1}^M \frac{(j+1)}{(M+1)^2}) = \frac{1}{M+1} + \frac{(M+1)(M+2)-2}{2(M+1)^3}$ , and otherwise player 1 will obtain strictly less.

Finally, we must remember that the pseudochance node imposes some restrictions on the game below it, and we have to check that these restrictions are satisfied. First, note that each of these subtrees are guaranteed to have a payoff of at least 1 for player 2, as player 2 has a choice of (0,1) at the top of each of these subtrees. Second, it should also be easy to see that in a satisfying assignment, that player 1's utilities will maintain the proper ordering. Finally, if player 1 acts to increase the utility of player 2 above 2, she can only increase it by  $\epsilon$ , and to do so, she sacrifices a large fraction of her utility at this subtree. Thus, there will exist an  $\epsilon$  where deviating in any subtree is counterproductive. This is enough to show that player 1's utility will be maximized in a satisfying assignment.  $\square$

### 3.2.5 Case 5. Three-players, DAG, pure or mixed strategies

If we allow for more than two players, solving for the optimal strategy to commit to in DAGs becomes NP-hard even when we only allow for pure strategy commitment.

**Theorem 14.** *It is NP-hard to solve for the optimal strategy to commit to in three-player games in DAG form, even in perfect-information games with no chance nodes and no costs/restrictions, regardless of whether commitment to mixed strategies is allowed.*

*Proof.* We reduce an arbitrary instance of SAT to an extensive-form game with the properties in the theorem, so that the leader can obtain utility 1 if the SAT instance is satisfiable and 0 otherwise. The SAT instance consists of  $N$  variables  $x_1, \dots, x_N$  and  $M$  clauses,  $C_1, \dots, C_M$ . The game is constructed as follows (an illustration is given in Figure 3.8).

#### Variable subtrees

For each variable  $x_i$ , we create a subtree  $t^{x_i}$  with a choice for player 1 between  $(0, 4, 0)$  and  $(0, 0, 4)$ . We will arrange the rest of the game so that player 1 committing to the pure strategy of playing  $(0, 4, 0)$  corresponds to choosing  $-x_i$  (setting  $x_i$  to *false*), and the pure strategy of playing  $(0, 0, 4)$  to  $+x_i$ . (We will discuss mixed strategies at the end of the proof.)

#### Clause subtrees

For each clause  $C_j$ , we create a subtree  $t^{C_j}$  with three levels. At the bottom level of this subtree is a node where player 2 acts, which is a parent of each variable subtree  $t^{x_i}$  where  $-x_i$  appears in  $C_j$ . (Note that the same variable subtree can descend from multiple clauses—this is why this is a reduction for DAGs and not for trees.) The node has one more child, corresponding to utilities of  $(0, 3, 3)$ .

At the middle level of this subtree is a node where player 3 acts, which is a parent of each variable subtree  $t^{x_i}$  where  $+x_i$  appears in  $C_j$ . The node has one more child, corresponding to the bottom level of the subtree.

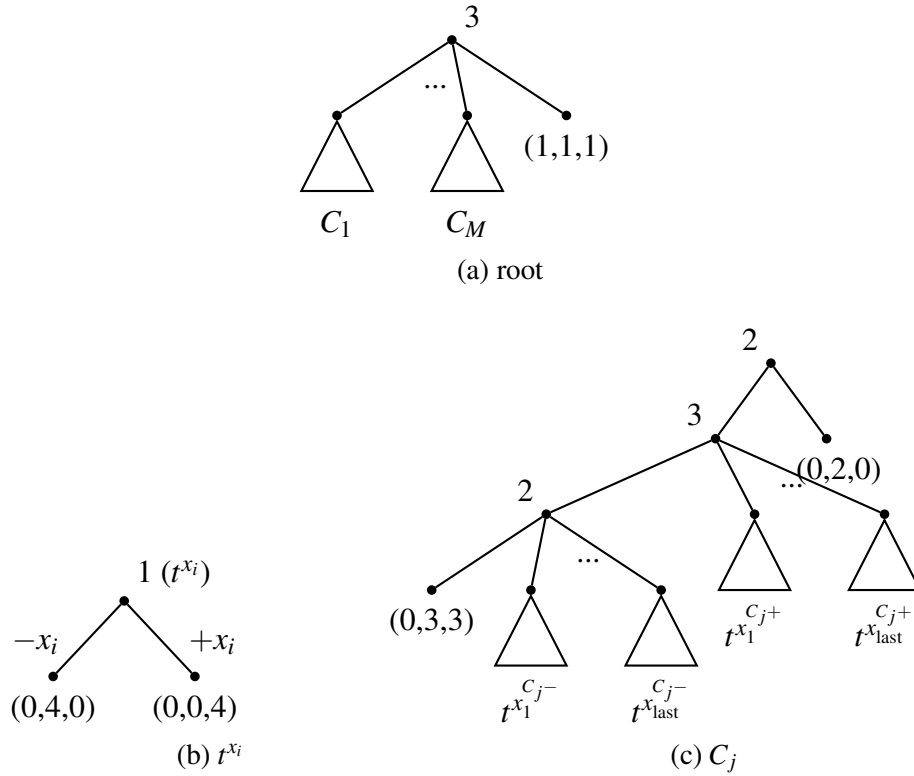


FIGURE 3.8: Three-player DAG used in the hardness reduction of Theorem 14. Note that  $t_{\text{last}}^{C_j^+}$  is the variable subtree corresponding to the last positive literal appearing in clause  $C_j$ .

At the root of this subtree is a node where player 2 acts, which is a parent of a child corresponding to the middle level, as well as a child with utilities  $(0, 2, 0)$ .

### Beginning of the game

Finally, at the first node (root) of the game, player 3 acts. This node is a parent of every clause as well as another node with utilities  $(1, 1, 1)$ .

### Proof of equivalence to SAT instance

Let us first consider the subtrees corresponding to clauses. The root of a clause subtree is a choice for player 2 between utilities  $(0, 2, 0)$  and continuing deeper into the subtree. In the latter case, if any positive literals of this clause are set to *true*, then player 3 will choose a corresponding variable subtree over continuing to the bottom level, because a

utility of 4 is more than the best that player 3 can expect from the bottom level, 3. In this case, player 2 would receive a utility of 0—so he will choose the  $(0, 2, 0)$  option at the top level, and hence player 3 will achieve a utility of 0 for this subtree. If none of the positive literals are set to *true*, but at least one of the negative literals is set to *true* (that is, its corresponding variable is set to *false*), then at the bottom level player 2 will choose this option over  $(0, 3, 3)$ . As a result, player 3 again cannot avoid achieving a utility of 0 in the subtree. Thus, in any subtree corresponding to a satisfied clause, player 3 can expect a utility of 0. In contrast, if the clause is not satisfied, then both players 2 and 3 will prefer the outcome of  $(0, 3, 3)$ , so that player 3 can expect a utility of 3 from that subtree.

Hence, in the first move of the game, if there is an unsatisfied clause, player 3 will move to that clause, resulting in a utility of 0 for player 1; otherwise, player 3 will choose the outcome  $(1, 1, 1)$ . This proves that with commitment to a pure strategy, player 1 can achieve a utility of 1 if and only if the SAT instance is satisfiable.

### **Extension to commitment to mixed strategies**

If we allow commitment to probabilities in the variable subtrees, we can define a variable to be set to *true* if it receives probability at least  $3/4$  on outcome  $(0, 0, 4)$ , and set to *false* if it receives under  $1/4$  on outcome  $(0, 0, 4)$ . Thus it is impossible to have a variable be both *true* and *false*; moreover, players 2 and 3 will still prefer the variable subtrees in the same cases as before. □

### *3.2.6 Case 6. Two-players, DAG, restrictions, pure or mixed strategies*

Finally, if we allow for restrictions, solving for the optimal commitment strategy in DAGs becomes NP-hard even in the two-player, pure-strategy commitment case.

**Theorem 15.** *It is NP-hard to solve for the optimal strategy to commit to in games in DAG form with commitment restrictions (or costs), even in two-player perfect-information games with no chance nodes, regardless of whether commitment to mixed strategies is allowed.*

*Proof.* We reduce an arbitrary instance of SAT to an extensive-form game with the properties in the theorem, such that the leader can obtain utility 1 if the SAT instance is satisfiable, and at most  $\varepsilon$  otherwise. The SAT instance consists of  $N$  variables  $x_1, \dots, x_N$  and  $M$  clauses,  $C_1, \dots, C_M$ . The game is constructed as follows (an illustration is given in Figure 3.9).

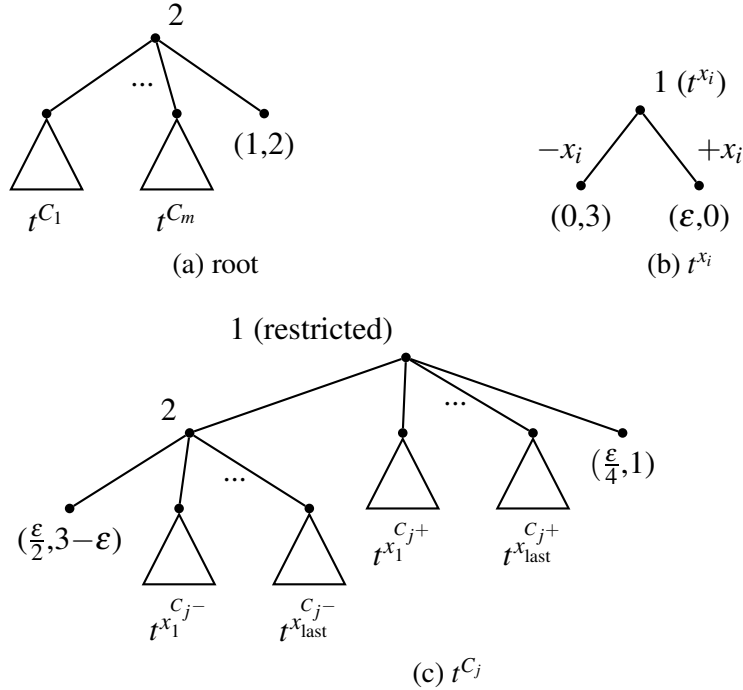


FIGURE 3.9: Two-player DAG with restrictions used in the hardness reduction of Theorem 15. Note that  $t^{x_{last}^{C_j^+}}$  is the variable subtree corresponding to the last positive literal appearing in clause  $C_j$ , etc.

### Top of the tree

The first move is by player 2, who can choose either an outcome of  $(1, 2)$ , or any one of the clause subtrees  $t^{C_j}$ , which we define next.

### Clause subtrees

For every clause  $C_j$ , there is a subtree  $t^{C_j}$ . In it, the first move is by player 1. This move is restricted (player 1 cannot commit to an action at this node, or, equivalently, the cost of commitment here is  $\infty$ ). There are three options for player 1 here: choose the outcome  $(\varepsilon/2, 1)$ ; choose any one of the subtrees  $t^{x_i}$  where  $+x_i \in C_j$  (these variable subtrees will

be defined shortly); or choose a node where player 2 moves. At this node where player 2 moves, he has two options: choose the outcome  $(\epsilon/2, 3 - \epsilon)$ ; or, choose any one of the subtrees  $t^{x_i}$  where  $-x_i \in C_j$  (these variable subtrees will be defined next).

### Variable subtrees

For every variable  $x_i$ , there is a subtree  $t^{x_i}$ . In this subtree, player 1 has two choices:  $(\epsilon, 0)$  and  $(0, 3)$ . Player 1 committing to choosing  $(\epsilon, 0)$  corresponds to setting variable  $x_i$  to *true* (so that  $+x_i$  is true) and committing to choosing  $(0, 3)$  corresponds to setting variable  $x_i$  to *false* (so that  $-x_i$  is true).

### Proof of equivalence to SAT instance

The only way for player 1 to obtain a utility of 1 is by incentivizing player 2 to choose the outcome  $(1, 3)$  from the root, instead of any clause subtree. Suppose that a clause  $C_j$  is satisfied because one of its positive literals  $+x_i$  is set to *true*. Then, in the first move in the subtree  $t^{C_j}$  (where commitment is not possible), player 1 will move to the corresponding subtree  $t^{x_i}$  to achieve the outcome  $(\epsilon, 0)$ , because further down in the subtree 1 can obtain at most  $\epsilon/2$ . Hence, player 2 will not prefer this clause subtree to  $(1, 2)$ .

Now, let us consider a clause  $C_j$  that is satisfied because one of its negative literals  $-x_i$  is set to *true* (meaning,  $x_i$  is set to *false*), but none of its positive literals is set to *true*. Then, if in the subtree  $t^{C_j}$ , player 2 gets to move, he will choose to move to the corresponding subtree  $t^{x_i}$  to achieve the outcome  $(0, 3)$ , which is better for player 2 than the outcome  $(\epsilon/2, 3 - \epsilon)$ . Hence, player 1 at the root of the clause subtree will choose the outcome  $(\epsilon/2, 1)$  (the subtrees corresponding to positive literals will all give her utility 0 as well). Hence, player 2 will not prefer this clause subtree to  $(1, 2)$ .

Finally, let us consider a clause  $C_j$  that is not satisfied. Both player 1 and player 2 prefer  $(\epsilon/2, 3 - \epsilon)$  to any other option, so this is the outcome that is reached. Hence, player 2 will prefer this clause subtree to  $(1, 3)$ .

Thus, if player 1 commits to pure actions in the variable subtrees, player 2 will choose  $(1, 3)$  if and only if player 1's commitment corresponds to a satisfying assignment. Next,

we extend this analysis to commitment to mixed actions by player 1.

### **Extension to commitment to mixed actions**

Finally, we argue that it is without loss of optimality for player 1 to commit to pure actions in the variable subtrees, so that the result still holds when player 1 is able to commit to mixed actions. The reason is that, to make player 2 prefer to choose a variable subtree  $t^{x_i}$  at his node in a clause tree  $t^{C_j}$  (with  $-x_i \in C_j$ ), player 1 needs to commit almost all her probability in this variable subtree to the left ( $-x_i$ ) branch. On the other hand, for player 1 to make herself prefer to choose a variable subtree  $t^{x_i}$  at her restricted node in a clause tree  $t^{C_j}$  (with  $+x_i \in C_j$ ), player 1 needs to commit at least 25% of her probability in this variable subtree to the right ( $+x_i$ ) branch. Hence, it is impossible to get both types of benefit from a variable subtree, and each individual type of benefit can be obtained using a pure commitment. □

## Commitment in Stochastic Games

A useful language for describing complex games that play out over time is that of *stochastic games*, a generalization of Markov decision processes (MDPs) to multiple players. Computing equilibria of stochastic games presents a number of challenges, but a recent sequence of papers makes significant progress on the problem of computing correlated equilibria that are not necessarily stationary (i.e., the players' actions may depend on the history of play, not just the current state). This line of research replaces the notion of value (the maximum utility) in traditional value iteration with an achievable set (the set of Pareto efficient maximal utilities for each player). Achievable sets represent all possible utility vectors yielded by correlated policies in equilibrium. Murray and Gordon (2007) presented the first exact algorithm for computing these achievable sets. However, the complexity of maintaining these achievable sets increased exponentially with each iteration, leading to an algorithm where both time and space requirements scaled exponentially. MacDermed et al. (2011) showed how to epsilon-approximate achievable sets efficiently while simultaneously lowering computational complexity, leading to the Quick Polytope Approximation of all Correlated Equilibria algorithm (QPACE).

This chapter is laid out as follows. In Section 4.1, we offer a short refresher on stochas-

		S		E		C	
		L	R	N		L	R
N	(0,1);E	(0,0);C	N	(0,0);E	U	(2,ε);E	(0,0);E
D	(0,0);E	(0,2);E					

FIGURE 4.1: Example stochastic game.

tic games. In Section 4.2 we discuss how the ability to send signals to the other player can change the outcome in stochastic games. In Section 4.3 we explore how much value a player might lose by either not being able to signal or not being able to commit. Next, in Section 4.4 we show that variants where signaling is not possible or the strategy can be based only on a finite amount of history are NP-hard. Finally, we show how QPACE can be easily modified to approximately compute commitment equilibria in Section 4.5. This chapter is based on Letchford et al. (2012).

## 4.1 Stochastic games

A two-player stochastic game is defined as follows: We have a two players, 1 and 2, a set of states  $T$  and in each state  $t$ , we have a set of actions  $A_t^i$  for each player  $i$ . For each state  $t$  and each action pair in  $A_t^1 \times A_t^2$ , we have an outcome that consists of two elements, the utilities that each of the players will achieve in that round and information on what state the game will transition to next (possibly stochastically). Finally, we have a discount factor  $\gamma$  which is used to discount the value of future payoffs.

Consider the game in Figure 4.1. This game has three states:  $S$ ,  $C$  and  $E$ . We assume state  $S$  is our initial state, meaning that play will begin with the possible actions  $A_S^1$  for player 1 and  $A_S^2$  for player 2 (throughout this chapter play will always begin in the state labeled  $S$ ). In this state player 1 has only 1 possible action; thus the outcome depends entirely on player 2. If player two chooses to play action  $L$  here (which we will denote

$L_S$ ), then the outcome is  $(0,1);E$ . This means that player 1 receives a utility of 0, player 2 receives a utility of 1, and the game transitions to state  $E$ , meaning in the next round the two players will be playing the normal-form game depicted as state  $E$ . Alternatively, if player 2 chooses  $R_S$ , then both players receive a utility of 0 for this round, and play transitions to state  $C$ .

## 4.2 Commitment and signals

In a two-player Stackelberg game, player 1 may be able to do more than just committing to play a specific mixed strategy; she may also be able to send signals to the other player. This idea has previously been explored for normal-form games [12]. Specifically, they consider the case where she can commit to drawing from a joint distribution over signals and her own actions, and then send the signal to player 2 before he moves, while playing her drawn action. (Without loss of generality, we can assume that the signal player 1 sends to player 2 is simply the action that he should take.) They show that in a two-player normal-form game, player 1 gains nothing from the ability to commit to such a correlated strategy rather than just a mixed strategy (that does not signal anything to the other player). The reason is that for each signal, there will be a distribution over player 1's actions conditional on that signal—and player 1 may as well just commit to playing the best of those distributions from her perspective.

However, there exists variations of correlated equilibrium for games that have an inherent time structure such as extensive form games. Von Stengel and Forges [63] propose a solution concept known as *Extensive Form Correlated Equilibrium*, where moves are only recommended to players when they reach the information set at which that move will be taken. Under a similar flexibility in signaling, it turns out that in two-player stochastic games, signaling about commitment becomes more meaningful. First, let us return to the game in Figure 4.1. If we assume  $\gamma = 1$ , if player 1 commits to with probability  $(.5 - \epsilon)$

		S				C				F				E	
		L	R			U	N			L	R			N	
N	(0,2);E	(0,0);F			D	(0,3);E	(0,0);E	N	(2,1);E	(0,0);C	N	(0,0);E			

FIGURE 4.2: Example game where signaling must occur early, but not too early.

send signal  $L_C$  and play  $U_C$  and commits to with probability  $(.5 + \varepsilon)$  send signal  $R_C$  and play  $D_C$ , then player 2 can expect a utility of  $1 + 2.5\varepsilon - \varepsilon^2$  for taking the action  $R_S$ . Without signaling, if player 1 commits to  $U_C$  less than  $\frac{2}{2+\varepsilon}$  of the time, player 2 will respond with  $R_C$  which leads to a utility of 0 for player 1. Furthermore, if player 1 commits to  $U_C$  at least  $\frac{2}{2+\varepsilon}$  of the time, player 2 will always prefer  $L_S$  which again gives a utility of 0 to player 1.

However, being able to signal about one's action at a state only when reaching that state may not be enough in every game. Consider the game pictured in Figure 4.2. To achieve a positive utility in this game, player 1 needs to signal what she will be playing in state  $C$  before player 2 chooses his action in state  $F$  but after he acts in state  $S$ . Consider what happens if player 1 commits to  $(.5 + \varepsilon)U_C + (.5 - \varepsilon)D_C$ . If she commits to sending the following signal to player 2 in state  $F$ :  $R_F$  when she will be playing  $U_C$  and  $L_F$  when she will be playing  $D_C$ , it is possible for the two players to achieve a correlated equilibrium of  $(.5 + \varepsilon)(R_F, U_C) + (.5 - \varepsilon)L_F$ . Given this, player 2 would prefer  $R_S$  to  $L_S$  as he would achieve an expected utility of  $2 + 2\varepsilon$  for  $R_S$ . In contrast, if player 1 only sends the signal after the transition to state  $C$ , then player 2 will prefer to play  $R_F$  (and  $L_S$ ). On the other hand, the information signaled informs player 2 of what action player 1 will play in state  $C$  and if this information is signaled too early, namely before player 2 makes a choice in  $S$ , then he would prefer to choose  $L_S$  when the signal is to play  $L_F$ . Thus, without the ability to signal this information at this specific time, this correlation would not be possible and player 1 would achieve a utility of 0.

### 4.3 Value of correlation and commitment

The game in Figure 4.2 illustrates a situation where player 1 would be unable to achieve a positive utility without both the ability to commit to a mixed strategy and the ability to correlate her actions with those of player 2. In this game commitment and correlation work synergistically. In this section we show it is not always so: in some games commitment by itself obtains all the value whereas correlation by itself obtains none, and in other games the situation is reversed.

Towards this end, we define the following three values. First, we define  $OptCom$  as player 1's utility in the optimal commitment strategy<sup>1</sup> that does not use correlation. Second, we define  $OptCor$  as player 1's utility in the correlated equilibrium that maximizes payoff for player 1. Finally, we define  $Opt$  as player 1's utility in the optimal commitment strategy that uses correlation.

	S		E	
	<i>L</i>	<i>R</i>	<i>N</i>	
<i>U</i>	$(\epsilon, 1); E$	$(1, 0); E$	<i>N</i>	$(0, 0); E$
<i>D</i>	$(0, 0); E$	$(1 - \epsilon, 1); E$		

FIGURE 4.3: Game where commitment offers an unbounded advantage over correlation.

**Theorem 16.** *There exists a stochastic game where  $\frac{OptCom}{OptCor} = \infty$  and  $OptCom = Opt$  for any discount factor.*

*Proof.* Consider the game in Figure 4.3.<sup>2</sup> This is effectively a one-shot game because of the immediate transition to an absorbing state. To calculate  $OptCom$  and  $OptCor$  for this game, we can reason as follows. In the normal-form game associated with  $S$ ,  $U_S$

<sup>1</sup>We use the standard definition of optimal here, where a commitment strategy is considered to be optimal if it maximizes the utility for player 1.

<sup>2</sup>We have used the stage game for state  $S$  elsewhere to point out that the value of commitment in normal-form games is  $\infty$  (Chapter 8).

dominates  $D_S$ , which allows us to solve this game by iterated strict dominance. Thus the only correlated equilibrium is  $(U_S, L_S)$  which gives a utility of  $\varepsilon$  to player 1. However, if player 1 is able to commit to playing  $D_S$  with probability 1, then player 2's best response to this is to play  $R_S$  (both with and without correlation). This causes player 1 to receive a utility of  $1 - \varepsilon$ . Thus,  $\frac{OptCom}{OptCor} = \frac{1-\varepsilon}{\varepsilon}$  which tends to  $\infty$  as epsilon approaches 0 and  $OptCom = Opt$ .

□

	S			E			C	
	L	R		N			L	R
U	(0,0);C	(0,.5);E		(0,0);E		U	(0,0);E	( $\frac{1}{\gamma}, \frac{\varepsilon}{\gamma}$ );E
D	(2\varepsilon,0);E	(\varepsilon,\varepsilon);E				D	( $\frac{\varepsilon}{\gamma}, \frac{1}{\gamma}$ );E	(0,0);E

FIGURE 4.4: Game where correlation offers an unbounded advantage over commitment.

**Theorem 17.** For any  $\gamma > 0$  there exists a stochastic game where  $\frac{OptCor}{OptCom} = \infty$  and  $OptCor = Opt$ .

*Proof.* Consider the game in Figure 4.4. To calculate  $OptCor$  and  $OptCom$  for this game, let us start by finding the optimal Stackelberg strategy without correlation. Let us first consider what is necessary to convince player 2 to play  $L_S$ . Even if player 1 commits to  $U_S$ , player 2 will best respond by playing  $R_S$  (which gives player 1 a utility of 0) unless player 1 commits to  $D_C$  with probability at least .5. If player 1 does commit to  $D_C$  with probability at least .5, then player 2's best response for  $C$  will be  $L_C$  and at best player 1 can expect at  $\varepsilon$  utility from  $C$ . Thus, there is no way for player 1 to achieve more than  $2\varepsilon$  by convincing player 2 to best respond with  $L_S$ . As both outcomes where player 2 plays  $R_S$  transition to the absorbing state  $E$  and give player 1 at most  $\varepsilon$  utility, we can conclude that player 1 will achieve at most  $2\varepsilon$  via commitment without correlation.

	$h = 0$	$0 < h < \infty$	$h = \infty$
Corr.	NP-hard(Th 18)	NP-hard(Th 19)	? <sup>3</sup>
No Corr.	NP-hard(Th 18)	NP-hard(Th 19)	NP-hard(Th 20)

FIGURE 4.5: Overview of hardness results.  $h$  represents the amount of history that player 1 can remember.

Next, consider the following correlated equilibrium, which involves:  $(U_S, L_S)$ ,  $.5(U_C, R_C) + .5(D_C, L_C)$ . This gives a discounted expected value of  $.5 + .5\epsilon$  for both players for state  $C$ , causing player 1 to prefer  $U_S$  to  $D_S$  and player 2 to prefer  $L_S$  to  $R_S$ . Player 1's utility under this equilibrium is  $.5 + .5\epsilon$ . Thus,  $\frac{OptCor}{OptCom} \geq \frac{.5+.5\epsilon}{2\epsilon}$  which tends to  $\infty$  as  $\epsilon$  goes to 0. □

#### 4.4 Hardness results

In this section we consider the difficulty of solving for an optimal Stackelberg strategy. We consider two main dimensions of the problem, the amount of memory about past states and actions player 1 can base her actions upon and the ability of player 1 to signal to the follower to enable correlation. For the memory dimension we consider three main cases. In the first case player 1 is constrained to commit to a stationary strategy. In the second case player 1 has some finite memory and can commit to act based upon the states and actions taken in these past timesteps. Finally, in the third case player 1 has an infinite memory, and can commit based on all actions and states that have occurred since the start of the game. For the signaling dimension we consider two cases, both with and without the ability to signal. An overview of our results appears in Figure 4.5.

**Theorem 18.** *It is NP-hard to solve for the optimal commitment to a stationary strategy in a stochastic game with or without correlation, for any discount factor  $\gamma > 0$ .*

*Proof.* We reduce an arbitrary instance of 3SAT to a stochastic game such that player 1 can obtain utility 1 if and only if the 3SAT instance is satisfiable. The 3SAT instance

---

<sup>3</sup>We show how to solve this approximately by a modification of the QPACE algorithm in Section 4.5.

S						
	N					
N	$(0,0); \frac{1}{M}C_1 + \dots + \frac{1}{M}C_m$					
E						
	N					
N	$(0,0);E$					
$C_i (x_1 \vee \neg x_2 \vee x_3)$						
	$C_i^{+x_1}$	$C_i^{-x_1}$	$C_i^{+x_2}$	$C_i^{-x_2}$	$C_i^{+x_3}$	$C_i^{-x_3}$
$C_i^{x_1}$	$(\frac{1}{\gamma}, 0); x_1$	$(0, 1); E$	$(0, 0); E$	$(0, 0); E$	$(0, 0); E$	$(0, 0); E$
$C_i^{x_2}$	$(0, 0); E$	$(0, 0); E$	$(\frac{1}{\gamma}, 0); E$	$(0, 0); x_2$	$(0, 0); E$	$(0, 0); E$
$C_i^{x_3}$	$(0, 0); E$	$(0, 0); E$	$(0, 0); E$	$(0, 0); E$	$(\frac{1}{\gamma}, 0); x_3$	$(0, 1); E$

FIGURE 4.6: Stochastic game used in the hardness reduction of Theorem 18.

consists of  $N$  variables  $x_1 \dots x_n$  and  $M$  clauses  $C_1 \dots C_m$ . The construction is pictured in Figure 4.6 and the details are as follows. We start with an initial state  $S$ , one state  $C_i$  for each clause, one state  $x_i$  for each variable and one final absorbing state  $E$ . Our initial state  $S$  has no payoff for either player, but transitions uniformly at random to a clause state  $C_i$ . Additionally,  $E$ , the absorbing state has a single possible outcome, namely  $(0,0);E$ .

**Clause states:** Each clause state is constructed as follows. For each literal we have one row ( $C_i^x$ ) and two columns ( $C_i^{+x}$  and  $C_i^{-x}$ ). If the literal is positive we have two entries in the game,  $(C_i^x, C_i^{+x})$  is assigned an outcome of  $(\frac{1}{\gamma}, 0); x$  and  $(C_i^x, C_i^{-x})$  is assigned an outcome of  $(0, 1); E$ . If the literal is instead negative, we include the following two entries,  $(C_i^x, C_i^{+x})$  is assigned an outcome of  $(\frac{1}{\gamma}, 0); E$  and  $(C_i^x, C_i^{-x})$  is assigned an outcome of  $(0, 0); x$ . The other 12 outcomes of the game are  $(0, 0); E$ . In Figure 4.6 we show an example for a clause with the literals  $(x_1 \vee \neg x_2 \vee x_3)$ .

**Variable states:** Each variable state has 1 column and two rows  $+x_i$  and  $-x_i$ . The outcomes are as follows, for row  $+x_i$  it is  $(0, \frac{1}{\gamma}); E$  and for row  $-x_i$  it is  $(0, 0); E$ .

**Proof of equivalence to 3SAT instance:** We now show that player 1 can obtain a utility

of 1 from this game if and only if there exists a satisfying assignment to the underlying 3SAT instance. Let us start by considering when player 1 can obtain a utility of 1 from a given clause state. We first consider a row that corresponds to a positive literal. In this case, if player 1 has also committed to  $+x$  in the corresponding variable state, then the two non-zero outcomes in this row give payoffs of  $(\frac{1}{\gamma}, 1)$  and  $(0, 1)$ . Since, by assumption, the follower breaks ties in player 1's favor, a signal to play  $C^{+x}$  (this could be either committing to play  $C^x$  or explicitly signaling this to player 2) will lead to a utility of 1 for player 1. If player 1 instead commits to any other strategy in  $x$  then a signal to play  $C^{+x}$  will instead lead to player 2 deviating to  $C^{-x}$  causing player 1 to receive 0 utility. Next, consider a row corresponding to a negative literal. In this case, if player 1 has committed to  $-x$  in the corresponding variable state, then the two potentially non-zero outcomes in this row gives payoffs of  $(0, 0)$  and  $(\frac{1}{\gamma}, 0)$ . With similar logic as before, a signal to play  $C^{-x}$  will lead to a utility of 0 for player 1 unless she has committed to  $-x$ . As there are three literals in each clause, this gives player 1 three potential ways to incentivize the follower to play in a way that is beneficial to player 1. However, later commitment (in the variable states) can remove this potential (namely commitment in such a way to preserve the potential for the opposing literal). If all three of these signals lose their potential, player 1 is left with no way to incentivize the follower to play in a way that gives her utility if play reaches this clause. Since the initial state forces a uniform random choice between these clauses subgames, this game will have expected value for player 1 of 1 if and only if all of the clause states have a utility of 1.  $\square$

**Theorem 19.** *It is NP-hard to solve for the optimal commitment to a strategy that uses a constant  $h$  steps of history with or without correlation and any discount factor  $\gamma > 0$ .*

*Proof.* Consider the following modification to the reduction used in the proof of Theorem 18. For each variable state, we insert  $h$  buffer states that give no payoffs before that state. Thus, by the time player 1 reaches the variable state, they will have forgotten which

clause they originated from and the above reduction will again hold.  $\square$

For the case of infinite history, it is impossible to extend the above 3SAT reduction. Consider the construction from Theorem 3 with  $h$  buffer states inserted for each variable. If player 1 has a memory of  $h + 1$ , when choosing a literal to commit to, player 1 can condition this upon the clause the players transitioned from. In this way, player 1 will be able to “satisfy” both the positive and negative values of each literal.

However, we note that stochastic games can model extensive-form games with chance nodes, which allows us to adapt the KNAPSACK reduction from Theorem 5 in [34] with minor changes.

**Theorem 20.** *It is NP-hard to solve for the optimal commitment in a stochastic game even when the strategy is allowed to use infinite history without correlation and any discount factor  $\gamma > 0$ .*

*Proof.* In the KNAPSACK problem, we are given a set of  $N$  items, and for each of them, a value  $p_i$  and a weight  $w_i$ ; additionally, we are given a weight limit  $W$ . We are asked to find a subset of the items with total weight at most  $W$  that maximizes the sum of the  $p_i$  in the subset. We reduce an arbitrary KNAPSACK instance to a stochastic game, in such a way that the maximal utility obtainable by player 1 with commitment (whether pure or mixed) is equal to the optimal solution value in the KNAPSACK instance. This game is illustrated in Figure 4.7, and defined formally below.

**Initial state S:** The first state contains two possible choices by player 2, who chooses between an outcome of  $(0, -W); E$  and a outcome which randomizes uniformly over the item states, defined next.

**Item states:** Each item  $I_i$  has two states. At the top level  $I_i^1$  (which can be reached directly from  $S$ ), there is a state where player 2 acts. It has two potential outcomes: one is an outcome of  $(\frac{Np_i}{\gamma}, \frac{-Nw_i}{\gamma}); E$ , the other is a transition to a state  $I_i^2$  where only player 1 has a choice. The latter node also has two outcomes:  $(0, \frac{-Nw_i}{\gamma^2}); E$  and  $(0, 0); E$ .

**Proof of equivalence to KNAPSACK instance:** If for an item  $i$ , player 1 commits to playing 100%  $\text{In}_{I_i^2}$ , then player 2, breaking ties in 1's favor, will move  $\text{In}_{I_i^1}$ , resulting in discounted payoffs of  $(Np_i, -Nw_i)$  if  $I_i^1$  is reached. Otherwise, player 2 will move  $\text{Out}_{I_i^1}$ , and player 1 will get 0 (and player 2 at most 0). Because player 1 wants player 2 to choose  $K_S$ , there is no benefit to player 1 in moving  $\text{In}_{I_i^2}$  with probability strictly between 0 and 100%, since this will only make  $K_S$  less desirable to player 2 without benefiting player 1. Thus, we can assume without loss of optimality that player 1 commits to a pure strategy.

Let  $X$  be the set of indices of states where player 1 commits to playing In. Then, player 2's expected utility for choosing  $K_S$  is  $(1/N) \sum_{i \in X} -Nw_i = -\sum_{i \in X} w_i$ . Player 2 will choose  $K_S$  if and only if  $\sum_{i \in X} w_i \leq W$ . Given this, player 1's expected utility is  $(1/N) \sum_{i \in X} Np_i = \sum_{i \in X} p_i$ . Hence, finding player 1's optimal strategy to commit to is equivalent to solving the KNAPSACK instance.  $\square$

S		$I_i^2$	
	K	A	
N	$\frac{1}{N}I_i^1 + \dots + \frac{1}{N}I_i^n$	$(0, -W); E$	N
			In
			Out
$I_i^1$		E	
	In	Out	N
N	$(\frac{Np_i}{\gamma}, \frac{-Nw_i}{\gamma}); E$	$(0, 0); I_i^2$	N
			(0, 0); E

FIGURE 4.7: Stochastic game used in the hardness reduction of Theorem 20.

## 4.5 Computing commitment equilibria

The QPACE algorithm [41] efficiently approximates the set of correlated equilibria in stochastic games by iteratively contracting a state's achievable set, by removing policies that violate a player's rationality constraints. Conitzer and Korzhyk (2011) showed that the set of commitment equilibria is equivalent to the set of correlated equilibria without the leader's rationality constraints. Therefore QPACE can be easily modified to approxi-

mately compute commitment equilibria by removing the leader rationality constraints in the achievable set contraction step.

Every iteration of QPACE performs a backup similar to single agent value iteration which improves the current estimation of each state's achievable set  $V(s)$ . Achievable sets are represented as polytopes with halfspace normals  $H_j$  and offsets  $V(s)_j$ . The normals  $H$  are fixed at initialization and are the same for all achievable sets. A state's backup is broken down into three steps: (1) Calculate the action achievable sets  $Q(s, \vec{a})$ , giving us the set of possible continuation utilities. (2) Construct a set of inequalities that defines the set of equilibria. (3) Approximately project this feasible set into value-vector space by solving a linear program for each hyperplane  $V(s)_j$ . Step two is the only step that needs to be changed to compute feasible commitment policies instead of correlated equilibria. We modify equation 6 in MacDermed et al. (2011) to not include rationality constraints for the leader. The resulting set of inequalities defining the set of commitment equilibria is shown in equation 4.1. This gives us our polytope in  $\mathbb{R}^{(n+1)|A|^n}$  over variables  $\vec{cu}_{\vec{a}i}$  and  $x_{\vec{a}}$  of feasible correlated equilibria:

For each player  $i$  who cannot commit,  
for each pair of distinct actions  $\alpha, \beta \in A_i$

$$\sum_{\vec{a} \in A^n} \vec{cu}_{\vec{a}(\alpha)i} \geq \sum_{\vec{a} \in A^n} x_{\vec{a}(\alpha)} [\vec{gt}_{\vec{a}(\beta)i} + R(s, \vec{a}^{(\beta)})_i] \quad (4.1)$$

$$\sum_{\vec{a} \in A^n} x_{\vec{a}} = 1 \text{ and } \forall \vec{a} \in A^n, x_{\vec{a}} \geq 0$$

For each joint-action  $\vec{a} \in A^n$  and halfspace  $j$

$$H_j \vec{cu}_{\vec{a}} \leq x_{\vec{a}} Q(s, \vec{a})_j$$

The rest of QPACE remains unchanged. Our modification to QPACE is minor and leaves the strong theoretical properties of the original algorithm intact. Most importantly, the algorithm converges to within  $\varepsilon$  in polynomial time and returns a set of commitment

equilibria which is guaranteed to include all exact equilibria with additional solutions being no worse than  $\varepsilon$ -equilibria, where  $\varepsilon$  is the approximation parameter.

## Solving Security Games on Graphs via Marginal Probabilities

We now switch our attention to the class of games described above as Stackelberg security games. Recall that Stackelberg security games often involve the allocation of multiple security resources to multiple targets (or, more generally, a single resource can sometimes be assigned to multiple targets simultaneously, in which case the subset is referred to as a *schedule*, where  $s \in S \subseteq 2^T$ ).

This added expressiveness causes general linear or mixed integer program formulations for the Stackelberg problem [13, 47, 64] to be exponential in size when applied to security games. A natural question to ask, however, is whether we really need to have a probability variable for every complete assignment of resources. Can we not restrict our attention simply to the *marginal* probability that a particular resource is assigned to a particular schedule? This would result in only a polynomial number of variables (one for each resource-schedule pair). In the context of security games, this approach was first pursued by Kiekintveld et al. (2009), showing that indeed it is possible to find polynomial-size formulations that only refer to the marginal probabilities. Unfortunately, there are examples where this approach fails, in that the marginal probabilities returned do not correspond

to any mixed strategy (distribution over assignments). On the other hand, Korzhyk et al. (2010) showed that the Birkhoff-von Neumann (BvN) theorem can be applied to show that under certain conditions, such a mixed strategy is guaranteed to exist and can be found efficiently. This work focused primarily on the relationship between the size of schedules, the applicability of the BvN theorem, and the complexity of computing Stackelberg strategies. For example, they showed that if schedules have size 1—so that resources are assigned to single targets—the BvN theorem applies, resulting in a polynomial-size linear program formulation for finding the marginal probabilities, together with an efficient algorithm for finding a corresponding mixed strategy.<sup>1</sup>

Our approach in this chapter is similar to that of Korzhyk et al. at a high level, but it turns out that for one of the cases we consider, we need a generalization of the BvN theorem that was given by Budish et al. (2013). We review a compact representation of security games and the Budish et al. result (which we give a brief overview of in Section 5.1). In Section 5.3 we use this generalization to characterize a new class of games where a mixed strategy corresponding to the marginal probabilities is guaranteed to exist and give an algorithm for finding it. In another case, we show how to compute optimal marginal probabilities in a specific way, and show how to directly obtain a mixed strategy that corresponds to these marginal probabilities. Finally, in Section 5.4 we show that in a number of settings there exists an instance where the optimal marginal solution does not correspond to any mixed strategy. Then, for most of these settings, we also show that the problem of finding an optimal Stackelberg strategy is in fact NP-hard. This chapter is based on Letchford and Conitzer (2013).

## 5.1 Background

### **Compact security game LP**

---

<sup>1</sup>A nonconstructive direct proof of the existence of a corresponding mixed strategy in this case was already given by Kiekintveld et al. (2009).

In the context of security games, the linear program presented in Section 1.2 has exponentially many variables  $p_\alpha$  (one for every assignment of resources). The following LP is an attempt to modify this LP to instead use variables for the *marginal* probabilities  $c_{\omega,s}$  that resource  $\omega$  is assigned to schedule  $s \in A(s)$ . (A similar LP was given in Korzhyk et al. (2010), which in turn was a linear program reformulation of the MIP given in Kiekintvelt et al. (2009).) In this LP,  $c_t$  denotes the probability that target  $t$  is covered.

Compact LP

$$\text{maximize } c_{t^*} U_d^c(t^*) + (1 - c_{t^*}) U_d^u(t^*)$$

subject to:

$$\forall \omega \in \Omega, \forall s \in A(\omega) : 0 \leq c_{\omega,s} \leq 1$$

$$\forall t \in T : c_t \leq \sum_{\omega \in \Omega, s \in A(\omega), t \in s} c_{\omega,s}$$

$$\forall t \in T : c_t \leq 1$$

$$\forall \omega \in \Omega : \sum_{s \notin A(\omega)} c_{\omega,s} = 0$$

$$\forall \omega \in \Omega : \sum_{s \in A(\omega)} c_{\omega,s} \leq 1$$

$$\forall t \in T : c_t U_a^c(t) + (1 - c_t) U_a^u(t) \leq c_{t^*} U_a^c(t^*) + (1 - c_{t^*}) U_a^u(t^*)$$

The problematic part of this LP is that it is not clear that it makes sense to set  $c_t = \sum_{\omega \in \Omega, t \in s \& s \in A(\omega)} c_{\omega,s}$ ; this calculation would be correct only if all the events where some  $\omega$  covers some  $s$  with  $t \in s$  are disjoint events. Since it is commonly assumed that the defender incurs no additional benefit for covering a target more than once, this LP actually provides an upper bound on the utility of the defender. Additionally, in some games, the optimal marginal probabilities  $c_t$  cannot be achieved in any mixed strategy. In previous work, the Birkoff-von Neumann [7] theorem has been used to characterize some special cases where it is guaranteed that the marginal probabilities do correspond to some mixed strategy [27]. In this chapter, we also have such a result, but we need a generalization of

the BvN theorem that we discuss next. (Also note that we allow  $c_t$  to be lower than the total probability on schedules that cover  $t$ ; this corresponds to the common assumption that every subset of a schedule is also a schedule, i.e., we can always reduce our coverage on a target without affecting anything else.)

### **Bihierarchy extension of BvN Theorem [8]**

For problems where there are two sets of objects  $X$  and  $Y$  (in our case, resources and schedules) and for every pair  $(x, y) \in X \times Y$  a marginal probability  $p_{x,y}$  (in our case,  $c_{\omega,s}$ ) is given, this result gives a sufficient condition for there to exist a distribution over assignments of  $X$  to  $Y$  (in our case, a distribution over assignments of resources to schedules, i.e., a mixed strategy) that is consistent with the marginal probabilities.

In these problems, generally, for various subsets  $Z \subseteq X \times Y$ , there is a constraint of the form  $\underline{q}_Z \leq \sum_{(x,y) \in Z} p_{x,y} \leq \overline{q}_Z$ , where  $\underline{q}_Z$  and  $\overline{q}_Z$  are integers. Not only does this constraint hold on the probabilities, but we also would like it to hold for the realized assignments; that is, in such a realized assignment  $\mu$ , we want  $\sum_{(x,y) \in Z} b_{x,y} \in \{\underline{q}_Z, \dots, \overline{q}_Z\}$ , where  $b_{x,y} \in \{0, 1\}$  indicates whether  $x$  is matched to  $y$  in  $\mu$ . (For example, in our problem, for each resource  $\omega$ , we want  $0 \leq \sum_{s \in A(\omega)} c_{\omega,s} \leq 1$ , and moreover in the assignments over which we randomize, we need  $\sum_{s \in A(\omega)} b_{\omega,s} \in \{0, 1\}$ .) The result [8] gives a sufficient condition on the family (or *constraint structure*)  $\mathcal{C}$  of sets  $Z$  (with  $Z \in \mathcal{C}$ ) to guarantee that a distribution over partial matchings that has the right marginal probabilities and that satisfies all the constraints can be found, namely that  $\mathcal{C}$  is a bihierarchy, defined as follows. A constraint structure  $\mathcal{H}$  is a *hierarchy* if, for every pair of elements  $Z, Z' \in \mathcal{H}$ , we have  $Z \subseteq Z'$ ,  $Z' \subseteq Z$  or  $Z \cap Z' = \emptyset$ . Constraint structure  $\mathcal{B}$  is a *bihierarchy* if there exists hierarchies  $\mathcal{H}_1$  and  $\mathcal{H}_2$  such that  $\mathcal{H}_1 \cup \mathcal{H}_2 = \mathcal{B}$  and  $\mathcal{H}_1 \cap \mathcal{H}_2 = \emptyset$ . [8] also gives an efficient flow-based algorithm for finding such a distribution.

HOMOGENEOUS RESOURCES				
<i>Graph \ Schedule</i>	Edge	Path (start root)	Path (pass through root)	Path (general)
Path	Yes/P (KCP 2010)	Yes/P (Th 21)	Yes*/P (Th 22)	Yes*/P (Th 22)
Tree	Yes/P (KCP 2010)	Yes/P (Th 21)	No/? (Ex 4)	No/NP-h (Ex 4/(LCL 2006))
General	No/P (KCP 2010)	No/NP-h (Ex 3/Th 24)	No/NP-h (Ex 4/Th 24)	No/NP-h (Ex 4/(LCL 2006))

HETEROGENEOUS RESOURCES				
<i>Graph \ Schedule</i>	Edge	Path (start root)	Path (pass through root)	Path (general)
Path	No/NP-h (Ex 2/Th 23)	Yes/P (Th 21)	?	No/NP-h (Ex 2/Th 23)
Tree	No/NP-h (Ex 2/Th 23)	Yes/P (Th 21)	No/NP-h (Ex 4/Th 25)	No/NP-h (Ex 4/(LCL 2006))
General	No/NP-h (Ex 2/Th 23)	No/NP-h (Ex 3/Th 24)	No/NP-h (Ex 4/Th 25)	No/NP-h (Ex 4/(LCL 2006))

FIGURE 5.1: Summary of results. Every entry states, before the /, whether the Birkhoff-von Neumann property holds (“Yes” or “No”), that is, whether marginal probabilities can always be realized. (“Yes\*” indicates that we only prove this for the marginal probabilities that result from our direct algorithm.) After the /, it states the complexity of finding an optimal Stackelberg strategy. All of the positive results hold even if the graph consists of multiple lines/trees/DAGs (that each have their own root in the rooted case), and all the negative results hold even if there is only one component.

## 5.2 Dimensions of the problem

In this section, we describe the different dimensions along which we vary the problem. The first dimension concerns whether the defender resources are homogeneous or heterogeneous. If we restrict ourselves to homogeneous defense resources, then any defender resource can cover any schedule.

The second dimension concerns the graph (whose vertices correspond to the targets and whose edges are related to the schedules). We consider:

- **Path graph.** A graph consisting of a single path.
- **Tree.** A graph with a single acyclic component.
- **General graph.** A general graph consisting of a single component.

In fact, for our positive results, we can also allow for multiple connected components in each case (e.g., forests instead of trees). We sometimes also require each component to have a distinguished root vertex.

The final dimension concerns how the graph restricts the possibilities for feasible schedules. We consider:

- **Edge.** Every schedule consists of two adjacent vertices in the graph.
- **Path.** Every schedule consists of a path in the graph. We also consider the further restrictions of: (1) only paths that pass through the root, and (2) only paths that start at the root.

We emphasize that not *every* subset of targets satisfying the requirement is a feasible schedule; rather only the converse is the case, that any subset *not* satisfying the requirement *cannot* be a feasible schedule. Our results are summarized in Figure 5.1.

**Motivating examples.** We now give two motivating examples for which we obtain positive results.

1. Consider a subway system without any cycles that is rooted at a central station. An inspector or guard can start a patrol from the central station, travel down a path, and then return along the same path. This corresponds to a tree graph with all feasible schedules being paths from the root.
2. Consider a high-speed rail line between two locations (such as the Japanese Shinkansen or the partially complete line between Beijing and Hong Kong). An inspector or guard can enter the train at some point, stay on it for some number of stops, and get off. This corresponds to a path graph with all feasible schedules being (sub)paths.

### 5.3 Positive results

In this section, we cover the cases where we can solve for an optimal strategy in polynomial time via marginal probabilities:

1. Defender resources are heterogeneous, the graph is a set of rooted trees, and schedules correspond to paths starting at a root.
2. Defender resources are homogeneous, the graph is a path graph, and schedules correspond to paths.

### 5.3.1 Heterogeneous, set of trees, paths from the root

**Theorem 21.** *There exists a polynomial-time algorithm for finding the optimal Stackelberg strategy when defender resources are heterogeneous, the graph is a set of rooted trees, and schedules correspond to paths starting at a root.*

*Proof.* The following constraints on the  $c_{\omega,s}$  variables hold:

1.  $\forall t \in T : \lfloor \sum_{\omega \in \Omega, s \in A(\omega) \& t \in s} c_{\omega,s} \rfloor \leq \sum_{\omega \in \Omega, s \in A(\omega) \& t \in s} c_{\omega,s} \leq \lceil \sum_{\omega \in \Omega, s \in A(\omega) \& t \in s} c_{\omega,s} \rceil$
2.  $\forall \omega \in \Omega : 0 \leq \sum_{s \in A(\omega)} c_{\omega,s} \leq 1$

The constraints under 2 clearly form a hierarchy, because their variable sets do not intersect with each other. When the graph is a set of rooted trees and schedules correspond to paths starting at a root, the constraints under 1 also form a hierarchy: if  $t$  is an ancestor of  $t'$  (i.e.,  $t$  is on the path from the root to  $t'$ ), then  $\{(\omega, s) : s \in A(\omega), t' \in s\} \subseteq \{(\omega, s) : s \in A(\omega), t \in s\}$ ; if neither of  $t$  and  $t'$  is an ancestor of the other, then  $\{(\omega, s) : s \in A(\omega), t' \in s\} \cap \{(\omega, s) : s \in A(\omega), t \in s\} = \emptyset$ . Therefore, these constraints form a bihierarchy, and we can apply the algorithm from [8] to obtain a probability distribution over (deterministic) assignments of resources to schedules, where in each such assignment:

- each resource is used at most once (by the first constraint),
- every target for which  $0 \leq \sum_{\omega \in \Omega, s \in A(\omega) \& t \in s} c_{\omega,s} < 1$  is covered either by 0 resources or by 1 resources, so that the coverage events are disjoint and the overall probability of covering  $t$  is in fact  $\sum_{\omega \in \Omega, s \in A(\omega) \& t \in s} c_{\omega,s}$ ,
- every target for which  $1 \leq \sum_{\omega \in \Omega, s \in A(\omega) \& t \in s} c_{\omega,s}$  is always covered by at least 1 resource.

We note that it is possible that  $c_t < \sum_{\omega \in \Omega, s \in A(\omega) \& t \in s} c_{\omega,s}$ ; this can cause a problem in the case of  $t = t^*$ , which we may wish to defend less in order to lure the attacker to

it. If so, we can simply move probability from a schedule  $s$  that places probability on  $t^*$  to schedule  $s' = s \setminus \{t^*\}$  (which is a feasible schedule under the subsets-of-schedules-are-schedules assumption), until  $c_{t^*} = \sum_{\omega \in \Omega, s \in A(\omega) \& t^* \in s} c_{\omega, s}$ . By the constraints of the Compact LP, the resulting strategy will incentivize the attacker to attack  $t^*$ , and obtain the optimal value from this LP.  $\square$

### 5.3.2 Homogeneous, path graph, general paths

**Theorem 22.** *There exists a polynomial-time algorithm for finding the optimal Stackelberg strategy when defender resources are homogeneous, the graph is a set of path graphs, and schedules correspond to (sub)paths.*

*Proof.* We start by solving the Compact LP above. However, for our argument, we will need a particular type of optimal solution for the marginal probabilities, which this LP is not guaranteed to give. The only parts of the LP solution that we will use are the attacker's preferred target  $t^*$  and the utility that the attacker receives in this solution ( $u_a = c_{t^*}^{\text{LP}} U_a^c(t^*) + (1 - c_{t^*}^{\text{LP}}) U_a^u(t^*)$ ). We now describe a method to generate a new marginal solution that results in the same utility for the defender as the LP solution and for which we can find a corresponding mixed strategy.

Let  $t_1, \dots, t_n$  be the targets in the order in which they appear in the path graph, from left to right. Initialize the current target coverage probabilities as  $c_{t_j} = 0$  for all  $j$ . Now, for  $j = 1$  to  $n$ , consider the schedule that covers  $t_j$  and extends as far as possible towards the right. If  $c_{t_j} U_a^c(t_j) + (1 - c_{t_j}) U_a^u(t_j) > u_a$ , then place on this schedule the amount of probability needed to make  $c_{t_j} U_a^c(t_j) + (1 - c_{t_j}') U_a^u(t_j) = u_a$ , and update the other targets' coverage probabilities accordingly. We then assign the probability mass that we have now placed on this schedule to the first remaining available resource. If the schedule requires more probability mass than this resource has left, then cover as much as possible with this resource, and cover the remainder with the start of the next resource. Let  $c_{\omega, s}$  be the probability mass from resource  $\omega$  that is assigned to schedule  $s$  at the end of the for loop.

It is straightforward to see that this solution minimizes the number of resources needed to bring the attacker down to utility  $u_a$ . Hence, this solution uses at most the number of defender resources used in the LP ( $m$ ).

Next, we need to guarantee that the attacker is incentivized to attack target  $t^*$ . If the attacker's expected utility for attacking target  $t^*$  is  $u_a$ , then we are done with this step. Otherwise, starting with the last schedule  $s$  in the marginal solution that covers  $t^*$ , we can simply move probability from schedule  $s$  to schedule  $s_0 = s \setminus \{t^*\}$  (which is a feasible schedule under the subsets-of-schedules-are-schedules assumption) until  $c_{t^*} U_a^c(t^*) + (1 - c_{t^*}) U_a^c(t^*) = u_a$ . Since our marginal solution now has the same coverage on target  $t^*$  as the LP solution, and guarantees that each other target's utility is at most  $u_a$ , it will give the defender the same expected utility.

Finally, we need to go from our marginal solution to a mixed strategy. The resulting mixed strategy can be intuitively described as follows. Randomly draw a single number  $p$  uniformly from  $[0, 1]$ . For each resource  $\omega$ , consider the schedule  $s$  that, in the sequential process described above, made  $\omega$ 's total assigned mass rise above  $p$ ; then, simply assign  $\omega$  to this schedule  $s$ . It is easy to see that this indeed results in a marginal probability of  $c_{\omega,s}$  that  $\omega$  is assigned to  $s$ .<sup>2</sup> All that remains to show is that a target either is never covered by more than one resource, or, if it sometimes is, then it is covered with probability 1. To show this, note that in the sequential process above, it is impossible that we add mass to a schedule containing resource  $t$ , then to a schedule not containing  $t$ , and then later again to a schedule containing  $t$ . Now, suppose that  $t$  is sometimes covered by two resources,  $\omega_1$  and  $\omega_2$ , where  $\omega_1$  is the earlier resource. Suppose this happens for  $p = p_0$ . Then, for

---

<sup>2</sup>Instead of performing this procedure explicitly when drawing an assignment, one can also write the mixed strategy simply as a mixture of pure strategies, as follows. Consider all values  $p_1, \dots, p_x \in [0, 1]$  at which one of the resources switches schedules. Then, for  $i \in \{1, \dots, x+1\}$ , assign probability  $p_i - p_{i-1}$  to the schedule that occurs when  $p_{i-1} < p < p_i$ , interpreting  $p_0 = 0$  and  $p_{x+1} = 1$ . We note that  $x+1$ , the total number of assignments with positive probability, can be at most  $n+1$ : in the sequential process above, every one of the  $n$  vertices introduces at most one new schedule, and the very first one does not correspond to a cutoff point, so at the end of the sequential process there were at most  $n$  cutoff points; however, in the process of reducing the probability on  $t^*$ , we may have introduced a single additional cutoff point by splitting a schedule  $s$  into  $s$  and  $s \setminus \{t^*\}$ , so  $x \leq n$ .

$p > p_0$ ,  $\omega_1$  must cover  $t$ ; and for  $p < p_0$ ,  $\omega_2$  must cover  $t$ . □

## 5.4 Negative results

In this section, we give our negative results. We consider the following settings:

1. Defender resources are heterogeneous, the graph is a path graph, and schedules correspond to edges.
2. Defender resources are homogeneous, the graph is general, and schedules correspond to paths from the root.
3. The graph is a tree and schedules correspond to paths through the root.

In each of these settings, we first give an example instance where the optimal marginal solution does not correspond to any mixed strategy. (In setting 3, this example works even for homogeneous resources.) Then, we show that the problem of finding an optimal Stackelberg strategy is in fact NP-hard. (In setting 3, we are only able to show this hardness for heterogeneous resources.)

### 5.4.1 *Heterogeneous resources, a path graph, schedules are edges*

**Example 2.** *Consider the graph depicted in Figure 5.2a. There are two defender resources, one that can cover either edge  $(A,B)$  or edge  $(C,D)$ , and one that can cover either edge  $(B,C)$  or edge  $(D,E)$ . Suppose the utility functions are such that the defender would like to defend  $A$  with probability  $1/2$ , and  $C$  and  $D$  each with probability 1. (For example, suppose that the defender greatly prefers  $A$  to be attacked rather than any other target, the attacker is not interested in  $B$  or  $E$ , and if  $A$  is defended with probability  $1/2$ , the attacker would prefer attacking  $C$  or  $D$  unless these are defended with probability 1.) The numbers in the figure depict a marginal solution that achieves this. However, it is easy to see that there is no mixed strategy that achieves this: whenever  $A$  is defended by the first*

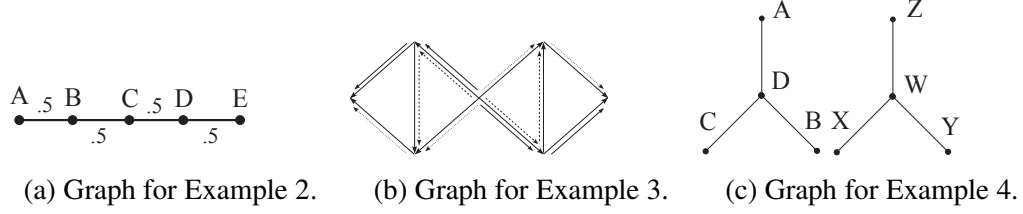


FIGURE 5.2

resource, it is impossible for the second resource to cover both  $C$  and  $D$ ; so, the latter two cannot be defended with probability 1 if  $A$  is defended with nonzero probability.

**Theorem 23.** *Finding the optimal Stackelberg strategy is NP-hard when defender resources are heterogeneous, the graph is a path graph, and schedules correspond to edges.*

*Proof.* We show this by a reduction from 3-COVER, in which we are given a set of elements  $\{1, 2, \dots, m\}$  and  $n$  subsets of size 3, and the goal is to find  $\frac{m}{3}$  sets that exactly cover all  $m$  elements.

**Structure of the resources, path graph, and schedules.**

We create one resource  $\omega_j$  for every element  $j \in \{1, 2, \dots, m\}$ . For every one of the  $n$  subsets  $S_i$ , we create three resources,  $\omega_{S_i}^1, \omega_{S_i}^2, \omega_{S_i}^3$ . Finally, we create  $n - m/3$  interchangeable resources  $\omega_\phi^1, \dots, \omega_\phi^{n-m/3}$ .

For the sake of exposition, we in fact create multiple path graphs; it is easy to connect all these together into a single path graph using dummy edges. For each subset  $S_i = \{j, k, l\}$ , we create 4 path graphs  $G_{S_i}^1, G_{S_i}^2, G_{S_i}^3, G_{S_i}^4$ . We use  $G_{S_i}$  to denote the union of these four graphs.  $G_{S_i}^1$  and  $G_{S_i}^4$  have three vertices and two edges;  $G_{S_i}^2$  and  $G_{S_i}^3$  have four vertices and three edges. The utility functions are such that our goal is to cover every internal vertex (with two neighbors) with probability 1; the leaf vertices do not need to be defended. Any resource  $\omega_\phi^x$  can defend the leftmost edge of  $G_{S_i}^1$ ; resource  $\omega_j$  can defend the leftmost edge of  $G_{S_i}^2$ ; resource  $\omega_k$  can defend the leftmost edge of  $G_{S_i}^3$ ; resource  $\omega_l$  can

defend the leftmost edge of  $G_{S_i}^4$ . Resource  $\omega_{S_i}^1$  can cover either the rightmost edge of  $G_{S_i}^1$  or the center edge of  $\omega_{S_i}^2$ ; resource  $\omega_{S_i}^2$  can cover either the rightmost edge of  $G_{S_i}^2$  or the center edge of  $\omega_{S_i}^3$ ; resource  $\omega_{S_i}^3$  can cover either the rightmost edge of  $G_{S_i}^3$  or the rightmost edge of  $\omega_{S_i}^4$ .

**Proof of equivalence** We now show that there exists a strategy that covers all the internal vertices with probability 1 if and only if there is a 3-cover. If there is a 3-cover, then consider the following pure strategy. If  $S_i = \{j, k, l\}$  is in the cover, then we defend the leftmost edges of  $G_{S_i}^2, G_{S_i}^3, G_{S_i}^4$  with resources  $\omega_j, \omega_k, \omega_l$ , respectively, and we defend the rightmost edges of  $G_{S_i}^1, G_{S_i}^2, G_{S_i}^3$  with resources  $\omega_{S_i}^1, \omega_{S_i}^2, \omega_{S_i}^3$ , respectively, thereby covering all the internal vertices of these path graphs. If  $S_i$  is not in the cover, then we defend the leftmost edge of  $G_{S_i}^1$  with some resource  $\omega_\phi^x$ , and we defend the middle edges of  $G_{S_i}^2, G_{S_i}^3$  and the rightmost edge of  $G_{S_i}^4$  with resources  $\omega_{S_i}^1, \omega_{S_i}^2, \omega_{S_i}^3$ , respectively, thereby covering all the internal vertices of these path graphs. Note that this uses exactly all the  $n - m/3$  resources  $\omega_\phi^x$ . Thus, there exists a strategy that covers all the internal vertices with probability 1.

Conversely, suppose that there exists a strategy that covers all the internal vertices with probability 1. Any pure strategy on which this mixed strategy places positive probability must cover all the internal vertices; consider one such pure strategy. Consider some subset  $S_i = \{j, k, l\}$  where this pure strategy does not allocate any of the resources  $\omega_\phi^x$  to  $G_{S_i}^1$ . To cover the center vertex of  $G_{S_i}^1$ ,  $\omega_{S_i}^1$  must be allocated to  $G_{S_i}^1$ . Then, to cover the left internal vertex of  $G_{S_i}^2$ ,  $\omega_j$  must be allocated to  $G_{S_i}^2$ . Then, to cover the right internal vertex of  $G_{S_i}^2$ ,  $\omega_{S_i}^2$  must be allocated to  $G_{S_i}^2$ . Then, to cover the left internal vertex of  $G_{S_i}^3$ ,  $\omega_k$  must be allocated to  $G_{S_i}^3$ . Then, to cover the right internal vertex of  $G_{S_i}^3$ ,  $\omega_{S_i}^3$  must be allocated to  $G_{S_i}^3$ . Then, to cover the center vertex of  $G_{S_i}^4$ ,  $\omega_l$  must be allocated to  $G_{S_i}^4$ . So, all of  $\omega_j, \omega_k, \omega_l$  have been assigned to  $G_{S_i}$  (and cannot be used elsewhere). Now, because there are only  $n - m/3$  resources  $\omega_\phi^x$ , there must be exactly  $m/3$  subsets  $S_i$  for which none of

the  $\omega_\phi^x$  resources are allocated to  $G_{S_i}^1$ . Moreover, these  $S_i$  cannot overlap on any element  $j$ , if they did, they would both have  $\omega_j$  assigned to them. Therefore, these  $S_i$  constitute a 3-cover.  $\square$

#### 5.4.2 Homogeneous, general graph, paths from the root

**Example 3.** Consider the graph depicted in Figure 5.2b. The root is the node in the middle. There are 6 feasible schedules, each corresponding to a path from the root. They correspond to the (solid, dashed, dotted) edges to the (left, right) of the root. For example, one schedule covers the center (root) node, the top-left node, and the far-left node (the solid path to the left).

We have 3 homogeneous resources that we can assign to these schedules. The marginal solution picks each of these six possible schedules with a probability of  $1/2$ , resulting in a total defense probability of 1 for each target. However, it is easy to see that it is not in fact possible to cover all the targets all the time: for any assignment, either the left or the right side of the graph has only one resource assigned to it, and therefore an uncovered target.

**Theorem 24.** Finding the optimal Stackelberg strategy is NP-hard when defender resources are homogeneous, the graph is general, and schedules correspond to paths starting at a single root vertex.

This can be shown by reduction from 3-COVER, however, we omit this proof due to space considerations.

#### 5.4.3 Tree graph, paths that pass through the root

For this case, our counterexample works even for homogeneous resources, but our NP-hardness result only works with heterogeneous resources; the complexity with homogeneous resources is open.

**Example 4.** Consider the graph depicted in Figure 5.2c. The root of each tree is the middle vertex. There are six feasible schedules, each consisting of a path from one leaf vertex to

another leaf vertex (covering both those leaves as well as the root of the corresponding tree). For example, one schedule covers targets  $A$ ,  $D$ , and  $B$ .

We have 3 homogeneous resources that we can assign to these schedules. The marginal solution picks each of these six possible schedules with a probability of  $1/2$ , resulting in a total defense probability of 1 for each target. However, it is easy to see that it is not in fact possible to cover all the targets all the time: for any assignment, either the left or the right tree has only one resource assigned to it, and therefore an uncovered target.

While this example uses two trees, it is easy to turn it into an equivalent single-tree example, by merging  $D$  and  $W$  into a single root vertex.

**Theorem 25.** *Finding the optimal Stackelberg strategy is NP-hard when defender resources are heterogeneous, the graph is a tree, and schedules correspond to paths that pass through the root.*

*Proof.* We show this by a reduction from tripartite matching, in which we are given sets  $X$ ,  $Y$ , and  $Z$  ( $|X| = |Y| = |Z| = n$ ) and a set of triples  $T \in X \times Y \times Z$ , and the goal is to find a set of  $n$  elements of  $T$ , no two of which have any elements in common (equivalently, that cover all the elements).

**Structure of the tree.** Start with a root vertex  $r$ . Directly connected to  $r$ , we have one vertex  $v_x$  for each element  $x \in X$  and one vertex  $v_y$  for each element  $v_y \in Y$ .

**Resources and schedules.** For each  $z \in Z$ , we have one resource  $\omega_z$ . For each triplet  $(x, y, z) \in T$ , we add the schedule  $\{v_x, r, v_y\}$  to  $A(\omega_z)$ .

### **Proof of equivalence**

We now show that there exists a perfect tripartite matching if and only if there exists a strategy that covers all of the targets with probability 1. If there is a perfect tripartite matching, then if  $(x, y, z)$  is in this matching, we let  $\omega_z$  defend  $\{v_x, r, v_y\}$ , to defend all the vertices. If a strategy exists that covers every target with probability 1, then any pure strategy in the support of this mixed strategy must cover every target; take any one such

pure strategy. Consider the set of all  $(x, y, z)$  such that resource  $z$  defends the schedule  $\{v_x, r, v_y\}$  in this pure strategy; this set must constitute a perfect tripartite matching.  $\square$

## Computing Optimal Security Strategies for Interdependent Assets

Many important settings exhibit interdependencies between potential targets of attack. These may be explicit, as in IT and supply chain network security, or implicit, as in defending critical infrastructure (where, for example, successful delivery of transportation services depends on a highly functional energy sector, and vice versa), or in securing complex software systems (with failures at some modules having potential to adversely affect other modules). In this chapter we offer a principled model of failure cascades that allows us to capture both the direct and indirect value of assets, and extend this model to capture uncertainty about the structure of the interdependency network.

We also consider a slightly different model of how the defender is constrained in assigning defense resources to targets. We relax the assumption that defense policies are costless, but resource bounded. Additionally, we also examine the assumption that security decisions amount to covering a set of targets, or not. While in numerous settings to which such work has been applied (e.g., airport security, federal air marshal scheduling) restricting the security to two possible states is very reasonable, in other settings one may

choose among many *security configurations* for each valued asset, each with differing effectiveness and expense. For example, in cybersecurity, protecting computing nodes could involve configuring anti-virus and/or firewall settings, with stronger settings carrying a benefit of better protection, but at a cost of added inconvenience, lost productivity, as well as possible licensing costs. Indeed, costs on resources may usefully replace resource constraints, since such constraints are often not hard, but rather channel an implicit cost of adding further resources. In Section 6 we show one way to model a spectrum of defense options.

In this chapter we first modify and extend the previous linear programming techniques for Stackelberg security games to allow for an arbitrary set of security configurations (rather than merely to cover a target, or not), as well as to account for both random and targeted failures, and replace hard constraints on defense resources with costs associated with specific security configurations in Section 6.1. We present and justify a crucial assumption on the nature of interdependencies that allows us to use our LP formulations which fundamentally assume independence between targets (Section 6.2.1). We then offer a simple model of interdependencies based on probabilistic failure cascades satisfying this assumption. Our model makes an explicit distinction between an intrinsic and indirect value of assets, the latter being due entirely to interdependencies. This allows an economically meaningful extension of a well-known independent cascade approach to modeling the spread of infectious diseases or ideas (Section 6.2). We demonstrate that for trees we can compute expected utilities for all targets in linear time (Section 6.2). We extend our model to capture uncertainty about network structure, and experimentally study the impact of such uncertainty (Sections 6.2.5 and 6.5). We show that our approach is both scalable to realistic security settings and offers much better solutions than state-of-the-art alternatives (Sections 6.3.2 and 6.3.2). Finally, we evaluate experimentally the properties of optimal defense configurations in real and generated networks (Section 6.4). This chapter is based on Letchford and Vorobeychik (2012).

## 6.1 Computing optimal randomized security configurations

While security games as described above naturally entail an attacker, most systems exhibit failures that are not at all a deliberate act of sabotage, but are due entirely to inadvertent errors. Even though such failures are generally far more common than attacks, the vast majority of work in security games posits an attacker, but ignores such failures entirely; essentially the lone exception is a paper by Zhuang and Bier [68] which offers an analytic treatment of a simple model making explicit the distinction between attacks and natural disasters. Our formulation below is, to our knowledge, the first to explicitly model both attacks and random failures in the Stackelberg security game literature.

To formalize, suppose that the defender can choose from a finite set  $O$  of security configurations for each target  $t \in T$ , with  $|T| = n$ . A configuration  $o \in O$  for target  $t \in T$  incurs a cost  $C_t^o$  to the defender. If the attacker happens to attack  $t$  while configuration  $o$  is in place, the expected value to the defender is denoted by  $U_d^o(t)$ , while the attacker's value is  $U_a^o(t)$ . A key assumption in Stackelberg security games is that the targets are completely independent: that is, player utilities only depend on the target attacked and its security configuration [25].

We revisit this assumption below when we turn to networked (interdependent) settings. We denote by  $c_t^o$  the probability that the defender chooses  $o$  at target  $t$ . Finally, let  $r$  be the prior probability of the defender that a failure will happen due to a deliberate attack. If no attack is involved, any target can fail; the defender's belief that target  $t$  randomly fails (conditional on the event that no attack is involved) is  $g_t$ , with  $\sum_t g_t = 1$ .

To adapt the previous formulations of Stackelberg security games to our domains of interest, we first modify the well-known multiple linear program (henceforth, multiple-LP) formulation that assumes target independence to incorporate an arbitrary set of security configurations, together with their corresponding costs of deployment. In the multiple-LP formulation, each linear program solves for an optimal randomized defense strategy

given that the attacker attacks a fixed target  $\hat{t}$ , with the constraint that  $\hat{t}$  is an optimal choice for the attacker. The defender then chooses the best solution from all feasible LPs as his optimal randomized defense configuration. The independence assumption becomes operational here because we treat defense configurations  $c_t^o$  for each target in isolation, as this assumption obviates the need to randomize over joint defense schedules for all targets. The LP formulation for a representative target  $\hat{t}$  is shown in Equations 6.1a-6.1d.

$$\begin{aligned} \max r \left( \sum_o U_d^o(\hat{t}) c_{\hat{t}}^{o,\hat{t}} \right) + (1-r) \left( \sum_{t,o} g_t U_d^o(t) c_{\hat{t}}^{o,\hat{t}} \right) \\ - \sum_t \sum_o C_t^o c_{\hat{t}}^{o,\hat{t}}. \end{aligned} \quad (6.1a)$$

s.t.

$$\forall_{o,t} c_{\hat{t}}^{o,\hat{t}} \in [0, 1] \quad (6.1b)$$

$$\forall_t \sum_o c_{\hat{t}}^{o,\hat{t}} = 1 \quad (6.1c)$$

$$\forall_t \sum_o U_a^o(t) c_{\hat{t}}^{o,\hat{t}} \leq \sum_o U_a^o(\hat{t}) c_{\hat{t}}^{o,\hat{t}} \quad (6.1d)$$

The intuition behind the multiple-LP formulation is that in an optimal defense configuration, the attacker must (weakly) prefer to attack *some* target, and, consequently, one of these LPs must correspond to an optimal defense policy.

Notice that we can easily incorporate additional linear constraints. For example, it is often useful to add a budget constraint of the form:

$$\forall_{\hat{t},t} \sum_o C_t^o c_{\hat{t}}^{o,\hat{t}} \leq B.$$

## 6.2 Incorporating network structure

### 6.2.1 A General Model of Interdependencies

Thus far, a key assumption has been that the utility of the defender and the attacker for each target depends only on the defense configuration for that target, as well as whether

it is attacked or not. In many domains, such as cybersecurity and supply chain security, assets are fundamentally interdependent, with an attack on one target having potential consequences for others. In this section, we show how to transform certain important classes of problems with interdependent assets into a formulation in which targets become effectively independent, for the purposes of our solution techniques.

Below we focus on the defender's utilities; attacker is treated identically. Let  $w_t$  be an *intrinsic worth* of a target to the defender, that is, how much loss the defender would suffer if this target were to be compromised with no other target affected (i.e., not accounting for indirect effects). In doing so, we assume that these worths are independent for different targets. Let  $s = \{o_1, \dots, o_n\}$  be the security configuration on all nodes. The probability that a given  $t'$  is affected depends on  $s$  and the target  $t$  chosen by the attacker. Let  $z_{s,t'}(t)$  be the marginal probability that target  $t'$  is affected when the attacker attacks target  $t$ . Assuming that the utility function is additive in target-specific worths and the attacker can only attack a single target, the defender's expected utility from choosing  $s$  when  $t$  is attacked is

$$U_d(t)[s] = E \left[ \sum_{t'} w_{t'} 1(t' \text{ affected} \mid s, t) \right] = \sum_{t'} w_{t'} z_{s,t'}(t),$$

where  $1(\cdot)$  is an indicator function. This expression makes apparent that in general  $U_d(t)[s]$  depends on defense configurations at all targets, making the problem intractable. We now make the crucial assumption that enables fast computation of defender policies by recovering inter-target independence.

**Assumption 1.** For all  $t$  and  $t'$ ,  $z_{s,t'}(t) = z_{o_t,t'}(t)$ .

In words, the probability that a target  $t'$  is affected when  $t$  is attacked only depends on the security configuration at the attacked target  $t$ . Below, we use  $o$  instead of  $o_t$  where  $t$  is clear from context.

A way to interpret our assumption is that security against external threats is not very efficacious once an attack has found a way into the system. Alternatively, if the utility of

nodes is derived from their contribution to overall connectivity (e.g., in communication networks, where removing a node can, for example, increase latency), it is quite natural to assume that removal of a node impacts global connectivity *regardless of security policies on other nodes*. Our assumption was also operational in other work on interdependent security [33], where a justification is through a story about airline baggage screening: baggage that is transferred between airlines is rarely thoroughly screened, perhaps due to the expense. Thus, even while an airline may have very strong screening policies, it is poorly protected from luggage entering its planes via transfers. Cybersecurity has similar shortcomings: defense is often focused on external threats, with little attention paid to threats coming from computers internal to the network. Thus, once a computer on a network is compromised, the attacker may find it much easier to compromise others on the same network.

Under the above assumption, the defender utility when  $t$  is attacked under security configuration  $o$  is:

$$U_d^o(t) = z_t^o(t)w_t + \sum_{t' \neq t} z_{t'}^o(t)w_{t'}.$$

By a similar argument and an analogous assumption for the attacker's utility, we thereby recover target independence required by the linear programming formulations above.

### 6.2.2 Cascading Failures Model

In general, one may use an arbitrary model to compute or estimate  $z_{t'}^o(t)$ . Here, we offer a specific model of interdependence between targets that is simple, natural, and applies across a wide variety of settings.

Suppose that dependencies between targets are represented by a graph  $(T, E)$ , with  $T$  the set of targets (nodes) as above, and  $E$  the set of edges  $(t, t')$ , where an edge from  $t$  to  $t'$  (or an undirected edge between them) means that target  $t'$  depends on target  $t$  (and, thus, a successful attack on  $t$  may have impact on  $t'$ ). Each target has associated with it a

worth,  $w_t$  as above, although in this context this worth is incurred only if  $t$  is affected (e.g., compromised, broken). The security configuration determines the probability  $z_t^o(t)$  that target  $t$  is affected if the attacker attacks it *directly* and the defense configuration is  $o$ . We model the interdependencies between the nodes as independent cascade contagion, which has previously been used primarily to model diffusion of product adoption and infectious disease [24, 17]; Mounzer et al. (2010) is a rare exception (a similar model was also proposed by Tsai et al. (2012), but involves both a defender and an attacker maximizing impact of information diffusion through cascades). The contagion proceeds starting at an attacked node  $t$ , affecting its network neighbors  $t'$  each with probability  $p_{t,t'}$ ; the contagion then spreads from the newly affected nodes  $t'$  to their neighbors, and so on. The contagion can only occur once along any network edge, and once a node is affected, it stays affected through the diffusion process. An equivalent way to model this process is to start with the network  $(T, E)$  and remove each edge  $(t, t')$  with probability  $(1 - p_{t,t'})$ . The entire connected component of an attacked node is then deemed affected.

### 6.2.3 Computing Expected Utilities

Given the independent cascade model of interdependencies between targets, we must compute expected utilities,  $U_d^o(t)$  and  $U_a^o(t)$ , of the defender and the attacker respectively (note that these are expectations only over the cascades, but not the defender's mixed strategy). In general, we can do so by simulating cascades starting at every node  $t$  (using breadth-first search), with expected utility of defender/attacker estimated as a sample average over  $K$  simulated cascades (expectation in this case is with respect to random realizations of attack success for specific targets as well as edges that become a part of the failure contagion). In several special cases, however, we can either compute these exactly and efficiently, or speed up utility estimation. We now address these special cases.

## Cascades on Trees

It is intuitive that when the dependency graph is a tree, expected utilities can be computed efficiently. A naive algorithm can do it in linear time *for each target*  $t$ , yielding quadratic time in total (since we must repeat the process for all targets). In fact, we can do it in linear time *for all targets*, as the following theorem asserts.

**Theorem 26.** *If  $(T, E)$  is an undirected tree we can compute expected utilities at targets in  $O(|T|)$  time.*

*Proof.* In this proof we use a shorthand  $z_t^o$  for  $z_t^o(t)$ . Let us define the neighbors of a target  $t$  as  $N_t$ . By definition, the expected utility of a given node  $t$  ( $U^o(t)$ ) is the direct utility at that node ( $z_t^o w_t$ ) plus the expected utility due to the cascading failure. The expected utility due to cascading failure is

$$z_t^o \sum_{t' \neq t} w_{t'} p(\text{failure}(t')|t)$$

where  $p(\text{failure}(t')|t)$  is the probability a node  $t'$  fails if node  $t$  fails. Since this is a tree, there is only one path between any pair of nodes, which means we can express  $p(\text{failure}(t')|t)$  as the product of probabilities of the edges on the path between  $t$  and  $t'$ .

Next, let us consider the set of paths generated by each pair of nodes in the tree. If we organize these paths by the edges they contain (and use linearity of expectation), we can express the expected utility of the contagion spreading across an edge  $(t, t')$ ,  $E[U[t, t']]$ , as:

$$E[U[t, t']] = p_{t, t'} \left( w_{t'} + \sum_{t'' \in N_{t'}, t'' \neq t} E[U[t', t'']] \right). \quad (6.2)$$

Thus, we can reason that for each node  $t$ :

$$U^o(t) = z_t^o U(t),$$

where

$$U(t) = w_t + \sum_{t' \in N_t} E[U[t, t']] \quad (6.3)$$

Now let us describe a two-pass algorithm for calculating  $U(t)$  for all  $t$ . First, choose an arbitrary node to be the root of the tree. In the first pass, we calculate the expected loss due to each edge from parent to child ( $E[U[P, C]]$ ) from the bottom of the tree upward. In the second pass, we calculate the expected loss on each edge from child to parent ( $E[U[C, P]]$ ) from the top of the tree downward. We can model this as a message passing algorithm, where calculating  $E[U[t, t']]$  is done by passing a message from  $t'$  to  $t$ . We can see by Equation 6.2 that the necessary inputs to calculate  $E[U_d[t, t']]$  are the messages from  $N_{t'} \setminus t$  to  $t'$ . We will now show that at the time that each of these messages is generated, all of the necessary inputs will be available.

Consider the edges between a given node  $t$  and its neighbors. Unless  $t$  is the root, one of these edges will be between  $t$  and its parent  $P_t$ , and the rest (possibly 0 in the case where  $t$  is a leaf node) will be between  $t$  and its children. Since in the first pass we are passing messages from child to parent and a node has only one parent, we will have received messages from  $N_t \setminus P_t$  when we generate the message from  $t$  to  $P_t$ .

For the second pass, when we pass information to a child of  $t$ ,  $C_t$ , we will have received messages from  $N_t$ , thus we again have the necessary information to generate the message from  $t$  to  $C_t$ .

Finally, once a node has received messages from all of its neighbors we can easily calculate the expected loss at each node by Equation 6.3. However, to achieve a runtime of  $O(n)$ , we need to be slightly more clever in how we store these values. By combining Equations 6.2 and 6.3 we can reason that  $E[U[t, t']] = p_{t,t'}(U(t') - E[U[t', t]])$ . This allows us to give an equivalent definition of  $U(t)$ :

$$U(t) = w_t + \sum_{t' \in N_t} p_{t,t'}(U(t') - E[U[t',t]]). \quad (6.4)$$

Now, consider the same two-pass algorithm as before, but rather than storing the expected loss for every edge, we merely store a running total of the expected loss at each node. We argue that by the same reasoning as before that the necessary calculations will have been performed before we need them as inputs. However, we still need to show that we can recover the correct value out of the values stored at the two nodes. When we calculate  $E[U[P,C]]$  in the first pass, the value stored at  $C$  will be  $(U(C) - E[U[C,P]])$ , since we have not yet updated  $C$  with  $E[U[C,P]]$ . However, when we reach this edge on the downward pass to calculate  $E[U[C,P]]$ ,  $P$  will have  $U(P)$  stored. Since the value stored at  $C$  is still  $(U(C) - E[U[C,P]])$ , we can easily calculate  $E[U[C,P]] = p_{C,P}(U(P) - E[U[P,C]]) = p_{C,P}(U(P) - p_{P,C}(U(C) - E[U[C,P]]))$  and update  $C$ .

Since we visit each edge twice, and perform a constant amount of work each time, we can bound the runtime by  $O(|E|)$ . Since in a tree  $|T| - 1 = |E|$ , we can also bound the runtime by  $O(|T|)$ .  $\square$

### *Cascades on Undirected Graphs*

In general undirected graphs, we can apply a very simple optimization in the way we sample cascades to obtain substantial speedups when the graph is dense. First, observe that rather than determining live edges as the cascade unfolds, we can instead flip the biased coin for each edge to determine whether it is live or not during a particular cascade *prior* to propagating the failure. The resulting graph contains a subset of edges from the original graph. At this point, observe that each potential target in a given connected component will result in the same defender/attacker utility. We therefore only need to compute the expected loss once for each connected component. When the size of the largest connected

component is  $O(|T|)$ , a likely scenario in dense graphs, this optimization results in an  $O(|T|)$  speedup.

#### 6.2.4 *The Significance of Capturing Interdependence*

An obvious question that may arise upon pondering the complexities of our framework is whether they are worthwhile: it may well be that previous approaches which assume target independence offer satisfactory approximation. We now show theoretically, and later experimentally, that our approach improves dramatically as compared to one which assumes independence.

**Proposition 2.** *There exists a family of problem instances for which the independence assumption yields a solution that is a factor of  $O(n)$  worse than optimal.*

*Proof.* Consider a star with edges all directed towards the spokes and cascade probabilities of 1. All nodes have worth  $w_t = 1$  for both defender and attacker. Now, suppose that cost of defending a target is  $e = 2$ . If targets are independent, it is clearly not worth defending any of them. But since they are, in fact, dependent, and the attacker will attack the hub, the utility of the defender is  $-1 * n = -n$ . If the defender recognizes the dependencies, she will protect the hub, and the total loss will be  $-3$ .  $\square$

#### 6.2.5 *Incorporating Uncertainty about the Network*

Applying our framework in real-world networked security settings requires an accurate understanding of the interdependencies. Thus far, we assumed that the actual network over which cascading failures would spread is perfectly known. A natural question is: what if our network model is inaccurate?

Formally, we model the uncertainty about the network as a parameter  $\varepsilon$  which represents the probability of incorrectly estimating the relationship between a pair of targets. Thus, if there is an edge between  $t$  and  $t'$ , we now let this edge be present with probability  $1 - \varepsilon$ . On the other hand, if  $t$  and  $t'$  are not connected in the graph given to us, we propose

that they are, in fact, connected with probability  $\varepsilon$ . Thus, when the graph is large, even a small amount noise will cause us to err about a substantial number of edges.<sup>1</sup>

Note that there is a natural way to incorporate this model of uncertainty into our framework. Let us interpret  $p_{t,t'}$  as the probability of a cascade from  $t$  to  $t'$  *conditional on an edge from  $t$  to  $t'$* . Then, if  $t$  and  $t'$  are connected, we modify cascade probabilities to be  $\hat{p}_{t,t'} = p_{t,t'}(1 - \varepsilon)$ , whereas if they are not connected, the cascade probability is  $\hat{p}_{t,t'} = p_{t,t'}\varepsilon$ .

### 6.3 Experiments

The goal of this section is to illustrate the value of our framework as a computational tool for designing security in interdependent settings. Specifically, we aim to demonstrate that our approach clearly improves on state-of-the-art alternatives, and offers a scalable solution for realistic security problems. We pursue this aim by randomly constructing dependency graphs using Erdos-Renyi (ER) and Preferential Attachment (PA) generative models [45], as well as using a graph representing a snapshot of Autonomous System (AS) interconnections generated using Oregon routeviews [59]; this graph contains 6474 targets and 13233 edges and thus offers a reasonable test of scalability. In the ER model every directed link is made with a specified and fixed probability  $p$ ; we refer to it as ER( $p$ ). The PA model adds nodes in a fixed sequence, starting from an arbitrary seed graph with at least two vertices. Each node  $i$  is attached to  $m$  others stochastically (unless  $i \leq m$ , in which case it is connected to all preceding nodes), with probability of connecting to a node  $j$  proportional to the degree of  $j$ ,  $d_j$ .

For the randomly generated networks, all data presented is averaged over 100 graph samples. Since we generate graphs that may include undirected cycles, we obtain expected utilities for all nodes on a given graph using 10,000 simulated cascades (below we show

---

<sup>1</sup>We assume here that both the defender and attacker share the same uncertainty about the network. An alternative model could consider an attacker that has more (or exact) information about the network. The resulting defender problem would become a Bayesian Stackelberg game.

that this is more than sufficient). Intrinsic worths  $w_t$  are generated uniformly randomly on  $[0, 1]$ . Cascade probabilities  $p_{t,t'}$  were set to 0.5 unless otherwise specified. In the sequel, we restrict the defender to two security configurations at every target, one with a cost of 0 which stops attacks with probability 0 and one with a cost of  $c$  which prevents attacks with probability 1.

### 6.3.1 Sampling Efficiency

Throughout our experiments we use 10,000 samples to evaluate the expected utilities of players. A natural question is: are we taking enough samples? To answer this, we systematically varied the number of samples between 0 (i.e., letting  $U^o(t) = -w_t$ ) and 100,000. Our results offer strong evidence that 10,000 samples is more than enough: the expected utility (evaluated using 100,000 samples) of the resulting defense configurations becomes flat already when the number of samples is 1000.

### 6.3.2 Scalability

An important question given the complexity of our framework is whether it can scale to realistic defense scenarios. To test this, we ran our framework on the AS graph consisting of 6474 targets and 13233 edges. Since this is a large undirected graph containing cycles, a sampling approach was required, but the total running time (including both sampling and solving linear programs) amounted to less than 1 hour. Given the importance of security, and the fact that *distributions* of security settings are computed once (or at least infrequently, as long as significant changes to the interdependency structure are not very frequent), this seems a relatively small computational burden.

### 6.3.3 Comparison to State-of-the-Art Alternatives

There are two prime computational alternatives to our framework. The first is to assume that targets are independent. While we showed above that in the worst case this can be

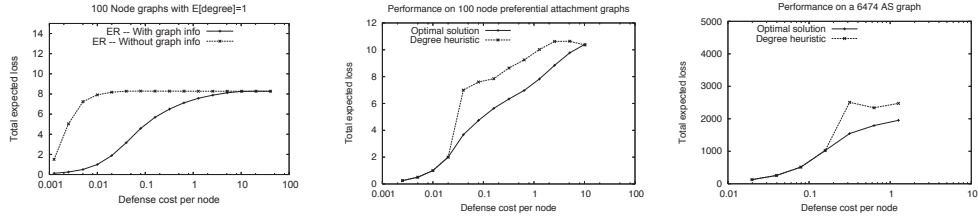


FIGURE 6.1: Left: Comparison between our approach (“with graph info”) and one assuming independence (“without graph info”) using the ER(0.1) generative model. Middle: Comparison to the degree-based heuristic on PA graphs. Right: Comparison to the degree-based heuristic on the AS graph.

quite a poor approximation, we offer empirical support to the added value of our approach below. The second is to use a well-known heuristic developed in the context of vaccination strategies on networks. This latter heuristic would in our case defend nodes in order of their connectivity (degree), until the defense budget is exhausted. Figure 6.1 compares our approach first to the former (left) and then to the latter (right). In both cases, computing optimal defense strategies using our framework yields much higher utility to the defender than the alternatives.

Aside from interdependencies, two other important aspects of our model are the fact that it allows an arbitrary number of security configurations, instead of simply allowing the defender to defend, or not, each target, and its ability to optimize with respect to both intelligent attackers and inadvertent failures. We now show that both of these can add substantial value. Figure 6.2 (left) shows a comparison between a solution which only allows two configurations (defend and do not defend) and two solutions which also allow for a third configuration, which is less effective than full defense, but also less costly. We consider two potential third options, one providing 50% defense at 12.5% of the cost of full defense ( $1/2 - 1/8$ ) and one providing 75% defense at 12.5% cost ( $3/4 - 1/8$ ). It is clear from this graph that considering the third configuration adds considerable value. Figure 6.2 (right) assumes that all (or nearly all) failures arise randomly, and compares a solution which posits an attacker to an optimal solution. Again, the value of solving the problem optimally is clear. This plot actually shows an interesting pattern, as the expected

utility of the defender is non-monotonic in cost when the solution is suboptimal. This is because the differences between the two solutions are most important when costs are intermediate; with low costs, nearly everything is fully defended, while high costs imply almost no defense.

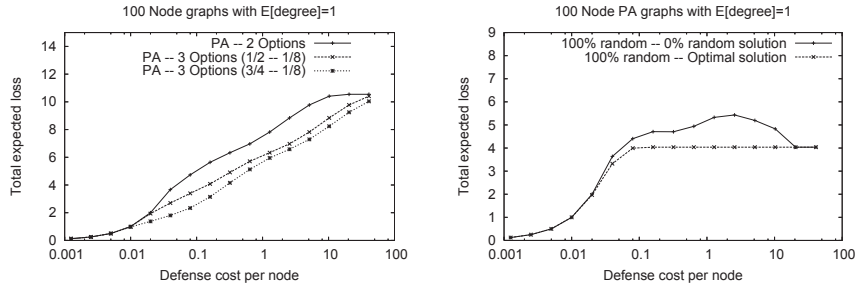


FIGURE 6.2: Left: Comparison between assuming only two configurations, and allowing the defender to consider three alternatives. Right: Comparison between a solution which assumes that failures are only due to attacks, and an optimal solution, when failures are actually random. Comparisons use PA graphs.

## 6.4 Applications to interdependent security analysis

In this section we apply our framework to several network security domains. For simplicity, we restrict attention to zero-sum security games, unless otherwise specified. As above, we consider ER and PA generative models, although we utilize a generalized version of PA. In a generalized PA model, connection probabilities are  $\frac{(d_i)^\mu}{\sum_j (d_j)^\mu}$ , such that when  $\mu = 0$  the degree distribution is relatively homogeneous, just as in ER,  $\mu = 1$  recovers the “standard” PA model, and large values of  $\mu$  correspond to highly inhomogeneous degree distributions. Throughout, we use  $\mu = 1$  unless otherwise specified. All parameters are set as in the experiments section, unless otherwise specified.

In addition to generative models of networks, we explore two networks derived from real security settings: one with 18 nodes that models dependencies among critical infrastructure and key resource sectors (CIKR), as inferred from the DHS and FEMA websites, and the second with 66 nodes that captures payments between banks in the core of the

Fedwire network [56].

For the CIKR network, each node was assigned a low, medium, or high worth of 0.2, 0.5, or 1, respectively, based on perceived importance (for example, the energy sector was assigned a high worth, while the national monuments and icons sector a low worth). Each edge was categorized based on the importance of the dependency (gleaned from the DHS and FEMA websites) as “highly” or “moderately” significant, with cascade probabilities of 0.5 or 0.1 respectively. For the Fedwire network, all nodes were assigned an equal worth of 0.5, and cascade probabilities were discretely chosen between 0.05 and 0.5 in 0.05 increments depending on the weight of the corresponding edges shown in [56].

## 6.5 The impact of uncertainty

Our framework offers a natural way to incorporate uncertainty about the network into the analysis. An important question is: how much impact on defender decision does uncertainty about the network have? Figure 6.3 quantifies the impact of uncertainty on the quality of defense if the observed graph is the PA network with average degree of 2. When cascade probabilities are relatively high ( $p_{t,t'} = 0.5$  for all edges, left plot), even if the amount of noise is relatively small ( $\epsilon = 0.01$ ), the resulting increase in the number of possible cascade paths in the network makes the defender much more vulnerable. With smaller cascade probabilities ( $p_{t,t'} = 0.1$ , right plot), however, noise has relatively little impact. It can thus be vital for the defender to obtain an accurate portrait of the true network over which failures may cascade when the interdependencies among the components are strong.

### 6.5.1 *The Impact of Marginal Defense Cost*

Our next analysis deals with the impact of marginal defense cost  $e$  on defender expected losses, her total costs, and the sum of these (i.e., negative expected utility). The results for ER and BA (both with 100 nodes and average degree of 2), as well as CIKR and Fedwire

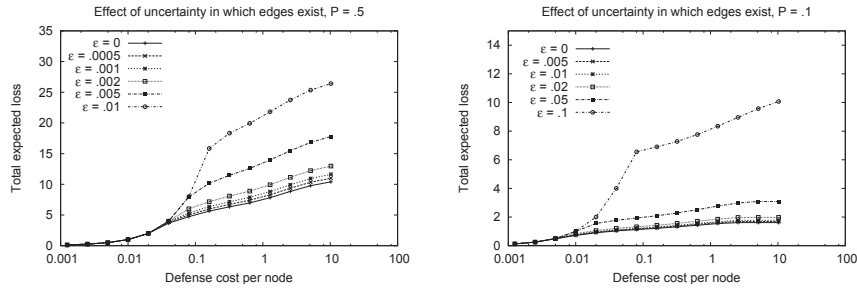


FIGURE 6.3: The impact of noise on PA networks. Left: when  $p_{t,t'} = 0.5$ ; Right: when  $p_{t,t'} = 0.1$ .

networks are shown in Figure 6.4. All the plots feature a clear pattern: expected loss and (negative) utility are monotonically increasing, as expected, while total costs start at zero, initially rise, and ultimately fall (back to zero in 3 of the 4 cases). It may at first be surprising that total costs eventually fall even as marginal costs continue to increase, but this clearly must be the case: when  $c$  is high enough, the defender will not wish to invest in security at all, and total costs will be zero. What is much more surprising is the presence of two peaks in PA and Fedwire networks. Both of these networks share the property that there is a non-negligible fraction of nodes with very high connectivity [45, 56]. When the initial peak is reached, the network is fully defended, and as marginal costs rise further, the defender begins to reduce the defense resources expended on the less important targets. At a certain point, only the most connected targets are protected, and since these are so vital to protect, total costs begin increasing again. After the second peak is reached,  $c$  is finally large enough to discourage the defender from fully protecting even the most important targets, and the subsequent fall of total costs is no longer reversed.

### 6.5.2 Resilience to Targeted Attacks: Impact of Network Structure

One of the important streams in the network science literature is the question of relative resilience of different network topologies to failures, random or targeted. A central result, replicated in a number of contexts, is that network topology is a vital factor in determining resilience [2, 45]. Of particular interest to us is the observation that scale-free networks

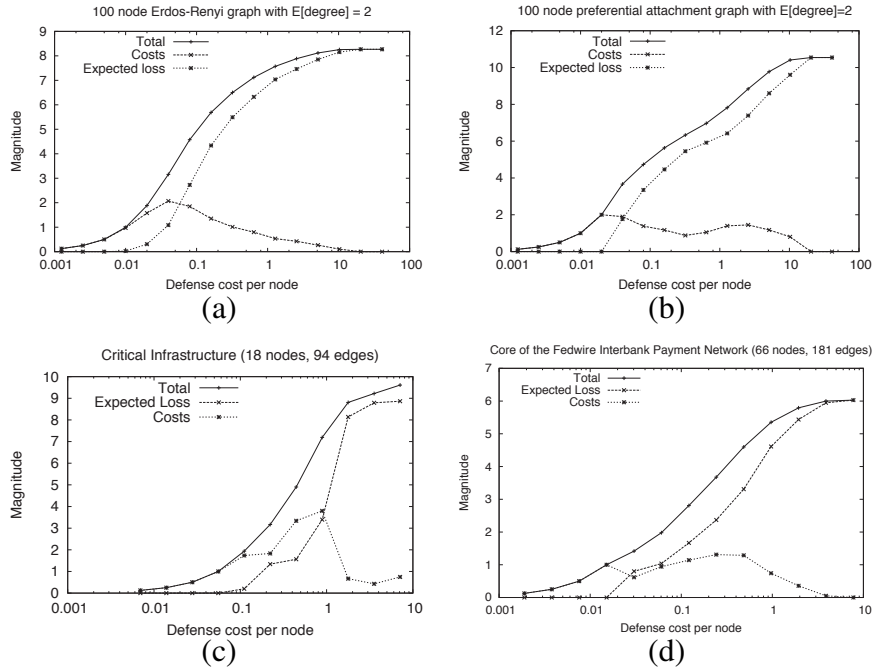


FIGURE 6.4: Expected loss, cost, and their sum in (a) 100-node ER(0.2), (b) 100-node PA, (c) 18-node critical infrastructure, and (d) 66-node core of the Fedwire networks as defense cost increases. The results for ER and PA are averages over 100 stochastic realizations of these networks.

such as PA exhibit poor tolerance to targeted attacks as compared to ER [2], which is precisely the context that we consider.

In Figure 6.5 (left) we show the defender’s utility for three different network topologies, PA, ER, and Fedwire as a function of cost  $e$ . Remarkably, there is essentially no difference between PA and ER (and not much between these and Fedwire) until  $e$  is quite high, at which point they begin to diverge. This seems to contradict essentially all the previous findings in that network topology seems to play little role in resilience in our case! A superficial difference here is that we consider a cascading failure model, while most of the previous work on the subject focused on diminished connectivity due to attacks. We contend that the most important distinction, however, is that previous work studying resilience did not account for a simple observation that most important targets are also most heavily defended; indeed, there was no notion of endogenous defense at all. In scale-free

graphs, there are well connected nodes whose failure has global consequences. These are the nodes which are most important, and are heavily defended in optimal decisions prescribed by our framework. Once the defense decision becomes endogenous, differences in network topology disappear. Naturally, once  $e$  is high enough, defense of important targets weakens, and eventually we recover the standard result: for high  $c$ , PA is considerably more vulnerable than ER.

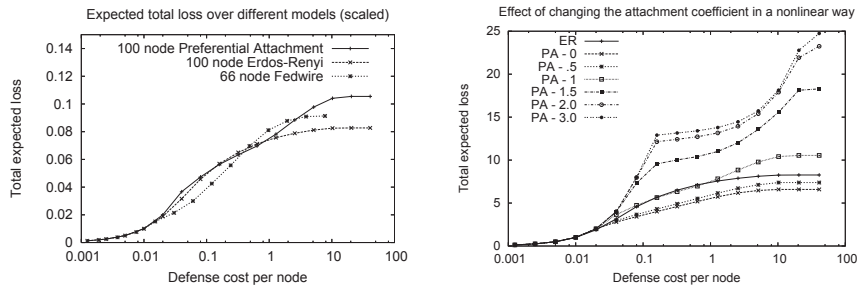


FIGURE 6.5: Left: Expected total loss: comparison across different network structures. Right: Expected defender disutility in the generalized PA model as we vary  $\mu$  (keeping average degree fixed at 2). ER is also shown for comparison.

To investigate the impact of network topology on resilience further, we consider the generalized PA model in which we systematically vary the homogeneity of the degree distribution by way of the parameter  $\mu$ . The results are shown in Figure 6.5 (right). In this graph, we do observe clear variation in resilience as a function of network topology, but the operational factor in this variation is homogeneity in the distribution of expected utilities, rather than degrees: increasing *homogeneity* of the utility distribution *lowers network resilience*. This seems precisely the opposite of the standard results in network resilience, but the two are in fact closely related, as we now demonstrate. Superficially, the trend in the figure seems to follow the common intuition in the resilience literature: as the degree distribution becomes more inhomogeneous (more star-like), it becomes more difficult to defend. Observe, however, that ER is actually more difficult to defend than PA with  $\mu = 0$ . The lone difference of the latter from ER is the fact that nodes that enter earlier are more connected and, therefore, the degree distribution in the PA variant should actually be more

*inhomogeneous* than ER! The answer is that random connectivity combined with inhomogeneity of degrees actually makes the distribution of *utilities* less homogeneous in PA with  $\mu = 0$ , and, as a result, fewer nodes on which defense can focus as compared to ER. On the other hand, as the graph becomes more star-like, the utilities of all nodes become quite similar; in the limiting case, all nodes are only two hops apart, and attacking any one of them yields a loss of many as a result of cascades.

There is another aspect of network topology that has an important impact on resilience: network density. Figure 6.6 shows a plot of an Erdos-Renyi network with the probability

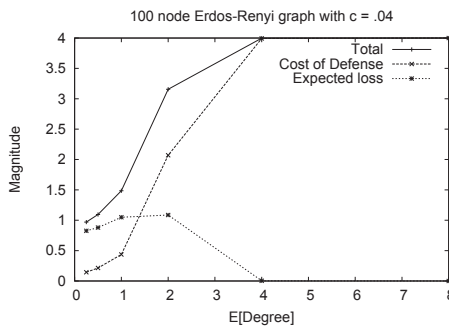


FIGURE 6.6: Expected loss, cost, and their sum in 100-node Erdos-Renyi networks as a function of network density (equivalently, expected degree).

of an edge varying between 0.0025 to 0.08 (average degree between .25 and 8) and cost  $e$  fixed at 0.04. Clearly, expected utility and loss of the defender are increasing in density, but it is rather surprising to observe how sharply they jump once average degree exceeds 1 (the ER network threshold for a large connected component); in any case, network density has an unmistakable impact. The reason is intuitive: increased density means more paths between targets, and, consequently, greater likelihood of large cascades in the event that a target is compromised. Total cost initially increases in response to increased density, in part to compensate for the increased vulnerability to attacks, but eventually falls, since it is too expensive to protect everything, and anything short of that is largely ineffective.

## Optimal Interdiction of Attack Plans

In this chapter we present a Stackelberg game model of security in which the defender chooses a mitigation strategy that interdicts potential attack actions, and the attacker responds by computing an optimal attack plan that circumvents the deployed mitigations. This may seem counter intuitive, as interdiction seems by its very nature an adversarial act, one perpetrated, if you will, by “bad guys”. For example, an attacker may interdict the power flow on an electric power grid, resulting in widespread blackouts [53], or interdict a transportation or a supply network [42, 20]. Indeed, even when the “good guys” engage in network interdiction, they do so with the goal of preventing activities, such as drug smuggling, by the criminals or adversaries. Conceptually, therefore, we deviate from that model: we argue that the nature of *defense* is fundamentally that of *interdicting attack plans*. In congruence with this point, we view the attacker as a planning agent that starts with a set of initial capabilities, and plans towards a set of specific goals. The defender’s goal is therefore to develop mitigation strategies that optimally interdict an attacker who actively seeks to circumvent deployed mitigations. In solving the interdiction problem, the defender takes into account both the cost of mitigation strategies, and their benefit in

terms of preventing the attacker from reaching a subset of goals. Again, our model is general-sum: the attacker and defender may have different priorities, and additionally, the attacker’s decision problem involves planning costs, while the defender is concerned with the cost of mitigations.

We begin by reviewing an integer programming approach for classical and partial satisfaction planning, which ultimately forms the core of our own framework in Section 7.1. Next, in Section 7.2 we formulate the optimal attack plan interdiction problem as a large-scale integer linear program, and offer several oracle-based approaches for solving it. We experimentally evaluate these oracle-based approaches in Section 7.3. In Section 7.4 we then proceed to generalize the model to capture two kinds of uncertainty: first, uncertainty about the attacker’s capabilities, costs, and goals, and second, uncertainty about whether a particular planning action is executed or not. Finally, in Section 7.4.3 we consider an attacker model in which the attacker is solving an MDP, and show how the corresponding interdiction problem can be formulated and solved. This chapter is based on Letchford and Vorobeychik (2013).

## 7.1 Integer programming for classical planning

While our starting point is the model of an attacker as a deterministic planner, we incrementally relax this assumption later on, attempting to retain as much of the classical planning structure (and relative tractability) in the process.

Formally, a classical planning problem is a tuple  $P = \{L, A, I, G\}$ , where  $L$  is the set of literals which capture all the information about the state of the world relevant for the planning problem,  $A$  is the set of actions,  $I$  is the set of literals which are initially true (i.e., the initial state of the world), and  $G$  is the set of goals. A plan action  $\alpha \in A$  is characterized by a set of *preconditions*, that is, the set of literals that must be true in the current state for the action to be applicable, and a set of *effects*, which are comprised of *add effects*,

or literals that are added to the state if  $\alpha$  is executed, and *delete effects*, which are the literals that  $\alpha$  deletes (i.e., makes false) after execution. Our starting point is actually an extension to the classical planning framework termed *partial satisfaction planning*, which assigns each goal literal a value, and each plan action a cost. Formally, we let  $U(l)$  denote the value of a goal literal  $l \in G$ , let  $U(l) = 0$  for all  $l \notin G$ , and let  $C^\alpha \geq 0$  denote the cost of action  $\alpha \in A$ .

The problem of finding an optimal plan given a fixed number of time-steps has a known integer programming (*IP*) formulation [65, 60] which will provide the basis for our own techniques. The objective of this *IP* is to find a plan which maximizes utility (i.e., value of achieved goals less plan execution costs):

$$\max \sum_{l \in L} U(l) s_l - \sum_{\alpha \in A} C^\alpha \sum_{t \in T} y_{\alpha,t}, \quad (7.1)$$

where  $y_{\alpha,t} = 1$  if and only if action  $\alpha$  is executed at time  $t$  and  $s_l = 1$  if and only if the literal  $l$  is satisfied by the computed plan. For our purposes below, this will be the *attacker's utility function*. The planning *IP* introduces the following set of meta-variables to capture how actions modify the state of each literal  $l$  at every time step  $t$ :

- $x_{l,t}^{pa}$  : 1 iff an action is executed in timestep  $t$  that has  $l$  as a precondition but does not delete it
- $x_{l,t}^{pd}$  : 1 iff an action is executed in timestep  $t$  that has  $l$  as a precondition and deletes it
- $x_{l,t}^{add}$  : 1 iff an action is executed in timestep  $t$  that does not have  $l$  as a precondition and adds it
- $x_{l,t}^{del}$  : 1 iff an action is executed in timestep  $t$  that does not have  $l$  as a precondition but deletes it

- $x_{l,t}^m$  : 1 iff  $l$  is true at timestep  $t - 1$  and no action in timestep  $t$  deletes it, adds it or has it as a precondition

The importance of these meta-variables is that they allow the set of constraints that enforce plan validity to be much more concise than if we were to reason about plan actions directly.

The constraints of the planning *IP* can be loosely categorized into three classes: constraints that build the above meta-variables, constraints that use these meta-variables to ensure that the computed plan is valid, and constraints that capture the initial conditions and goals. As an example, computing  $x^{pa}$  requires the following constraints:

$$\forall_{l,t} \sum_{\alpha \in pre_l \setminus del_l} y_{\alpha,t} \geq x_{l,t}^{pa} \quad (7.2a)$$

$$\forall_{l,t, \alpha \in pre_l \setminus del_l} y_{\alpha,t} \leq x_{l,t}^{pa} \quad (7.2b)$$

where  $pre_l$  represents the set of actions which have as  $l$  a pre-condition and  $del_l$  the set of actions where  $l$  is deleted as a post-condition. Each of the other meta-variables is computed with a similar set of constraints.

Enforcing plan validity amounts to ensuring that first, no mutually exclusive actions (e.g., actions that both add and delete a literal) are executed in the same time step, and second, that the state of a literal only changes at time  $t$  if there is an action executed at time  $t$  which either adds or deletes it. Formally, these constraints are:

$$\forall_{l,t} x_{l,t}^{pa} + x_{l,t}^m + x_{l,t}^{pd} \leq x_{l,t-1}^{add} + x_{l,t-1}^{pa} + x_{l,t-1}^m \quad (7.3)$$

$$\forall_{l,t} x_{l,t}^{pa} + x_{l,t}^m + x_{l,t}^{pd} \geq x_{l,t-1}^{add} \quad (7.4)$$

$$\forall_{l,t} x_{l,t}^{pa} + x_{l,t}^m + x_{l,t}^{pd} \geq x_{l,t-1}^{pa} \quad (7.5)$$

$$\forall_{l,t} x_{l,t}^{pa} + x_{l,t}^m + x_{l,t}^{pd} \geq x_{l,t-1}^m \quad (7.6)$$

The final set of constraints establish the initial conditions and calculate which goal literals

are satisfied by the plan:

$$\forall_l x_{l,|T|}^{add} + x_{l,|T|}^{pa} + x_{l,|T|}^m \geq s_l \quad (7.7)$$

$$\forall_l x_{l,|T|}^{add} \leq s_l \quad (7.8)$$

$$\forall_l x_{l,|T|}^{pa} \leq s_l \quad (7.9)$$

$$\forall_l x_{l,|T|}^m \leq s_l \quad (7.10)$$

$$\forall_l x_{l,0}^{add} = \begin{cases} 1 & \text{if } l \in I \\ 0 & \text{otherwise} \end{cases} \quad (7.11)$$

To make the problem more concrete, consider the following example of an attack planning problem.

**Example 5.** *The initial state includes the initial attacker capabilities, such as possession of a boot disk and port scanning utilities. Actions include both physical actions (breaking and entering and booting a machine from disk) as well as cyber actions, such as performing a port scan to find vulnerabilities.<sup>1</sup> Figure 7.1 shows an attack graph (with attack actions as nodes) for this scenario, with the actual attack plan highlighted in red.*

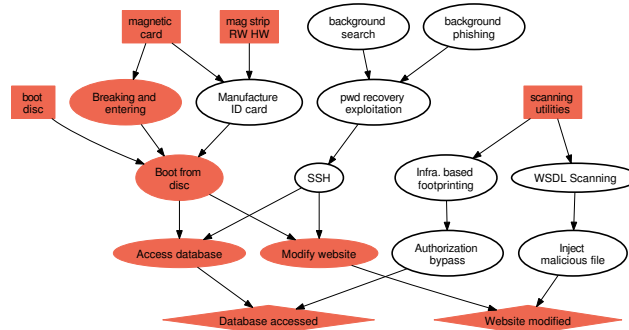


FIGURE 7.1: Example attack graph. Boxes correspond to initial attacker capabilities, ovals are attack actions, and diamonds are attacker goals. An optimal attack plan is highlighted in red.

<sup>1</sup>The actions in our example are taken from the CAPEC database (<http://capec.mitre.org>).

## 7.2 Deterministic plan interdiction

### 7.2.1 The Plan Interdiction Problem

Our ultimate goal is not merely to compute an optimal plan for the attacker, but rather to compute *an optimal defender interdiction strategy*. To this end, we model the interaction between the defender and attacker as a Stackelberg game in which the defender moves first, choosing to deploy a set of mitigations, and the attacker responds to these by constructing an optimal attack plan given the resulting environment. We formalize this game as a *deterministic plan interdiction problem (DPIP)*. *DPIP* is described by a tuple  $\{P, M, U_d(l), U_a(l), C_d^m, C_a^\alpha\}$ , where  $P$  is the planning problem for the attack in the absence of any mitigations, as described in Section 7.1,  $M$  is the set of mitigation strategies for the defender,  $U_d$  and  $U_a$  are the utilities of the defender and attacker respectively when the attacker achieves a goal literal  $l \in G$ ,  $C_d^m$  is the cost of mitigation  $m \in M$  to the defender, and  $C_a^\alpha$  is the cost of action  $\alpha \in A$  to the attacker. A mitigation strategy  $m \in M$  has two effects: it can protect against a subset of attack actions  $\alpha$  (effectively removing them from  $A$ ), and it can remove literals from the initial state  $I$ . Without loss of generality, we assume that  $m$  only has an effect on attacker actions (we can model removal of initial state literals by having actions with no preconditions, zero cost, and with  $l \in I$  as the only add effect). For each mitigation  $m \in M$ , let  $A_{m,\alpha} = 1$  iff mitigation  $m$  removes action  $\alpha$ . The defender's goal is to maximize her utility:

$$\max \sum_{l \in L} U_d(l) s_l - \sum_{m \in M} D_m C_d^m, \quad (7.12)$$

where  $D_m = 1$  iff mitigation  $m$  is chosen by the defender. The key complication is that, just as in general Stackelberg games, the defender's utility is a function of the attacker's best response, i.e., the optimal attack plan in response to the choice of defense mitigations. This raises the technical issue of tie breaking: if the attacker is indifferent between multiple plans, which would he choose? A common solution concept for Stackelberg se-

curity games is a Strong Stackelberg equilibrium, in which the attacker breaks ties in the defender's favor (we call this *optimistic tie breaking*). While counterintuitive, this solution concept is reasonable when the defender can commit to a randomized strategy, since she can resolve the attacker's indifference in her favor with an infinitesimal change in strategy. As we restrict the defender to choose mitigations deterministically, this justification becomes problematic, and it may be most reasonable to focus on the Weak Stackelberg equilibrium, that is, having the attacker break ties to minimize defender's utility (we call this *pessimistic tie breaking*). Below, we consider both variants and show empirically that in the context of deterministic commitment the difference between them is negligible in practice.

Before we delve into algorithmics, we proceed to answer the question of hardness of *DPIP*. First, we formulate this as a decision problem (*DPIP* decision problem, or *DPIPDP*).

**Definition 1** (*DPIPDP*). *Given a DPIP, is there a subset of mitigations  $M^* \subset M$  that yields defender a utility of at least  $k$ ?*

**Theorem 27.** *DPIPDP is PSPACE-Complete under both optimistic and pessimistic tie breaking, even when all mitigation strategies remove only a single action.*

*Proof.* First, let us show that *DPIPDP* is in *PSPACE*. Consider that there are at most  $|A|$  actions that player 1 must decide to defend or not. Additionally, for each defense strategy there exists in response an optimal plan for the attacker that has at most  $2^{|A|}$  total actions. Any optimal plan of length greater than  $2^{|A|}$  must have loops, and as all costs are non-negative, there must be an equivalent plan that requires at most  $2^{|A|}$  actions. Thus, as at most  $|A|2^{|A|}$  nondeterministic choices are required so *DPIPDP* is in *NPSPACE*. As  $NPSPACE = PSPACE$ , *DPIPDP* is also in *PSPACE*.

To show that *DPIPDP* is *PSPACE*-hard, we reduce from an arbitrary instance of PSP Net Benefit with  $P = (L, A, I, G)$  and  $k$  (as defined above). To do this we create an instance

of *DPIPDP*,  $P' = (L', A', I', G')$ , as follows: First, we create  $L'$  by adding three literals  $l_{(choice)}$ ,  $l_{(PSP)}$  and  $l_{(alt)}$  to  $L$ . Next, we create  $A'$  by adding two actions,  $\alpha_{(PSP)}$  and  $\alpha_{(alt)}$  to  $A$ .  $\alpha_{(PSP)}$  has as a precondition  $l_{(choice)}$  and has as postconditions  $\overline{l_{(choice)}}$  and  $l_{(PSP)}$ .  $\alpha_{(alt)}$  has as a precondition  $l_{(choice)}$ , has as postconditions  $\overline{l_{(choice)}}$  and  $l_{(alt)}$ . Attacker costs for actions in  $A$  are unchanged and  $C_a^{\alpha_{(PSP)}} = C_a^{\alpha_{(alt)}} = 0$ . As  $P$  does not model defender cost, we assign this in  $P'$  as follows: for each action  $\alpha_i \in A$   $C_d^{\alpha_i} = 2$ ,  $C_d^{\alpha_{(PSP)}} = 2$  and  $C_d^{\alpha_{(alt)}} = .5$ . Next, we add  $l_{(PSP)}$  as a precondition to every action in  $A$ . We create  $I'$  by adding  $l_{(choice)}$  to  $I$ . Finally, we create  $G'$  by adding  $l_{(PSP)}$  and  $l_{(alt)}$  to  $G$ . Attacker utilities for  $l \in G$ , but  $U_a(l_{(alt)}) = k$  and  $U_a(l_{(PSP)}) = U_a(l_{(choice)}) = 0$ .  $P$  again does not model defender utility, we assign this in  $P'$  as follows:  $\forall l \neq l_{(PSP)} U_d(l) = 0$  and  $U_d(l_{(PSP)}) = 1$ .

### *Proof of equivalence*

First, consider the case where there exists a plan in  $P$  that gives the attacker a utility of at least  $k$ . Since  $\alpha_{(PSP)}$  and  $\alpha_{(alt)}$  are mutually exclusive, the attacker will choose  $\alpha_{(PSP)}$ . If the defender has chosen to not defend  $\alpha_{(alt)}$  she will receive a utility of 1. However, if she chooses to defend  $\alpha_{(alt)}$  she will incur a cost of .5 and will obtain a net utility of .5.

In the case where no such plan exists, if  $\alpha_{(alt)}$  is undefended, then the attacker will prefer  $\alpha_{(alt)}$  and the defender will receive a utility of 0. If  $\alpha_{(alt)}$  is defended, then the defender will again receive a utility of .5.

Thus when the defender can achieve an optimal utility of 1 in  $P'$ , then there exists a plan in  $P$  that gives net benefit of at least  $k$ .

### *Pessimistic case*

To modify the above construction for the pessimistic case, we make the following changes. First, we modify defender costs for  $\alpha_{(PSP)}$  and  $\alpha_{(alt)}$ ,  $C_d^{\alpha_{(PSP)}} = .5$  and  $C_d^{\alpha_{(alt)}} = 2$ . Additionally, we change defender utility for literals  $l_{(alt)}$  and  $l_{(PSP)}$ ,  $U_d(l_{(alt)}) = 1$  and  $U_d(l_{(PSP)}) = 0$ .

The analysis is similar to the optimistic case, except when there exists a plan in  $P$  that gives the attacker a utility of at least  $k$ , the defender must defend  $\alpha_{(PSP)}$  and will be able to achieve a utility of .5. In the case where no such plan exists, the defender’s optimal strategy will be to not defend, and she will receive a utility of 1. Thus, when the defender’s optimal utility in  $P'$  is .5, there exists a plan in  $P$  that gives a net benefit of at least  $k$ .  $\square$

### 7.2.2 Integer Programming Formulation for Optimal Plan Interdiction

Despite the hardness result above, we now proceed to show how to compute optimal interdiction strategies in practice, and later experimentally demonstrate that our approaches are effective. For the moment, we ignore the issue of tie breaking, and will return to it in Section 7.2.5.

Our first step is to formulate the *DPIP* as a (very large; more on that later) integer program. The objective is, naturally, to maximize the defender’s utility (Equation 7.12). The complication is that we must capture the attacker’s best response to the defender’s choice of mitigations. We do so by constructing a special set of constraints that amounts to ensuring that: a) the plan that the integer program chooses for the attacker is feasible (with mitigations imposing appropriate feasibility constraints), and b) this plan has a payoff to the attacker that is at least as high as any other feasible plan. The way we approach this is to consider (for now) the set of all possible attacker plans  $\mathcal{P}$ . Clearly, the optimal plan  $p^*$  is in  $\mathcal{P}$ . Moreover, we can use the set of constraints described in Section 7.1 to compute a feasible plan  $p$ , as well as its utility to the attacker. Thus, if we further ensure that the utility of  $p$  computed in the constraints is at least that of *any* feasible plan in  $\mathcal{P}$ , subject to the additional constraints imposed by the mitigations, we know that  $p$  is in fact the attacker’s best response. To formalize this, let  $U_a(p)$  be the (precomputed) attacker utility of a plan  $p$ , and let  $\delta_p = 1$  if and only if plan  $p$  is interdicted (i.e., there is a deployed mitigation  $m$  that removes at least one action from  $p$ ). Let  $Z$  be a large constant. The

complete integer program for  $DPIP$ , which we call  $DPIP\_IP$ , is given by

$$\max_{D_\alpha, D_m, y_{\alpha,t}, \delta_p} \sum_{l \in L} U_d(l) s_l - \sum_{m \in M} D_m C_d^m \quad (7.13)$$

s.t. :

$$\forall_\alpha \quad D_\alpha \leq \sum_m D_m A_{m,\alpha} \quad (7.14)$$

$$\forall_{m,\alpha} \quad D_\alpha \geq D_m A_{m,\alpha} \quad (7.15)$$

$$\forall_{\alpha,t} \quad y_{\alpha,t} \leq (1 - D_\alpha) \quad (7.16)$$

$$\forall_{p,\alpha} \quad \delta_p \geq D_\alpha \quad (7.17)$$

$$\forall_p \quad \delta_p \leq \sum_{\alpha \in p} D_\alpha \quad (7.18)$$

$$\forall_p \quad \sum_{l \in L} U_a(l) s_l - \sum_{\alpha,t} C_a^\alpha y_{\alpha,t} \geq U_a(p) - Z \delta_p \quad (7.19)$$

constraints on metavariables (Sec. 7.1)

constraints 7.3 – 7.6, 7.7 – 7.11.

(Note that we use  $p$  here both as a plan index and as a set of actions that make up this plan). Constraints 7.14 and 7.15 compute a variable  $D_\alpha$  which is 1 iff there is a mitigation that interdicts action  $\alpha \in A$ . Constraint 7.16 ensures that only actions which are *not* interdicted are used in the attack plan. Constraints 7.17 and 7.18 compute  $\delta_p$ . Finally, Constraints 7.19 ensure that the plan computed for the attacker is his best response to the defender's mitigations.

### 7.2.3 Scaling Up with Constraint Generation

The main problem with  $DPIP\_IP$  is that the number of feasible plans and, consequently, the number of constraints, is exponential in the number of actions. To manage this problem in practice we develop several constraint generation (Bender's decomposition) approaches [6].

First, consider a relaxed version of *DPIP\_IP* with Constraints 7.14-7.19 corresponding to a *subset*,  $\hat{\mathcal{P}}$ , of all possible plans. We call this *master* problem *DPIP\_MASTER*( $\hat{\mathcal{P}}$ ). Now, suppose that we solve the master problem, obtaining a set of mitigations  $\hat{M} \subset M$  and that *DPIP\_MASTER*( $\hat{\mathcal{P}}$ ) identifies  $\hat{p} \in \hat{\mathcal{P}}$  with a utility of  $\hat{U}$  as the attacker's best response from the plans restricted to  $\hat{\mathcal{P}}$ . There are now two possibilities: either  $\hat{p}$  is the best response of the attacker to  $\hat{M}$ , or the true attacker best response is not in  $\hat{\mathcal{P}}$ . To see which it is, we need to compute the actual best response of the attacker; fortunately, that is just the standard planning problem, and we can use the integer program from Section 7.1 upon removing the actions  $\alpha \in A$  that are blocked by mitigations  $\hat{M}$ . The plan that we thereby compute will either have a utility to the attacker that is exactly  $\hat{U}$ , confirming that  $\hat{M}$  is the optimal solution (since  $\hat{p}$  is a true best response), or strictly higher. In the latter case, we add the newly computed plan to the master program, and repeat. This procedure is presented in Algorithm 1.

---

**Algorithm 1:** Constraint generation algorithm.

---

```

 $\hat{\mathcal{P}} = \emptyset;$ 
 $\hat{U} = 0;$ 
 $U = \infty;$ 
while  $\hat{U} < U$  do
     $(\hat{M}, D_\alpha, \hat{U}) = \text{DPIP\_MASTER}(\hat{\mathcal{P}});$ 
     $A_{\hat{M}} = \emptyset;$ 
    for  $\alpha \in A$  do
        if  $D_\alpha = 0$  then
             $A_{\hat{M}} = A_{\hat{M}} \cup \alpha;$ 
        end
    end
     $(p, U) = \text{optimalPlan}(A_{\hat{M}});$ 
    if  $U > \hat{U}$  then
         $\hat{\mathcal{P}} = \hat{\mathcal{P}} \cup p;$ 
    end
end

```

---

To see that Algorithm 1 converges to the optimal interdiction strategy in finite time,

consider what happens in any given iteration: either a new plan is added to the master program, or we prove that we have already computed an optimal solution. Since the total number of plans is finite—even if extremely large—the number of iterations must be finite. Therefore, the algorithm is sound (since we are generating best responses in each iteration) and complete.

One more subtle but practically consequential point (as we shall see in the experiments). While the initial incarnation of the master program is missing the Constraints 7.14-7.19 altogether, the set of constraints is still non-trivial, as it ensures that the attacker’s response to mitigations is *feasible* (and most favorable for the defender). Therefore, the initial set of mitigations will in general be non-empty, and will already interdict some of the attack actions. In other words, even this initial master program can be viewed as making non-negligible progress towards the goal.

**Example 6.** *We applied our method to the problem in Example 5, allowing the defender to interdict each attack action at a cost. The solution for a somewhat arbitrary assignment of parameters is shown in Figure 7.2.*

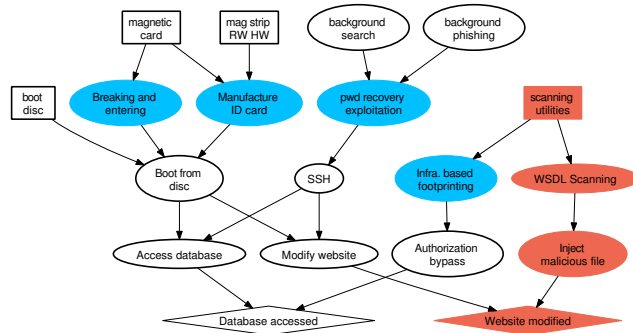


FIGURE 7.2: Example interdiction plan: actions that are blocked are colored in blue, and the final attack plan (circumvention) is highlighted in red.

#### 7.2.4 Optimistic Constraint Generation

Generating constraints for the master problem using an integer program from Section 7.1 clearly “works”, in the sense that we will typically need to generate a relatively small set

of plans (as we show in the experiments below). However, observe that to make progress in each iteration of Algorithm 1 we need only to generate a plan with a higher utility than any in  $\mathcal{P}$ , and not necessarily an optimal plan. Doing so may, of course, result in more iterations, but if we can sufficiently speed up each iteration, on balance we could have a win.

There are two natural candidates for such optimistic constraint generation. The first is to still use the planning *IP* from Section 7.1, but to cut off the solver after a fixed time limit. The second is to leverage the extensive heuristic planning work in AI, and use an off-the-shelf state-of-the-art heuristic planner, such as SGPLAN [10].

If we use optimistic constraint generation, Algorithm 1 is no longer sound, as we may generate a suboptimal plan which would make a current result appear optimal, even though it is not. To ensure soundness, we must therefore check the final solution obtained using a planner which does guarantee optimality, such as the planning *IP* from Section 7.1. However, since this needs only to be done relatively rarely (often only at the very end), most iterations will still run much faster than they would otherwise.

### 7.2.5 Tie breaking

The approach above breaks ties somewhat arbitrarily, since only a single best response is ever generated. Below we show how to handle pessimistic tie breaking and, thus, compute a Weak Stackelberg equilibrium; we can deal with optimistic tie breaking analogously.

Our first step is to ensure that the defender's utility is at most that of any non-interdicted attack plan *currently generated*. To that end we introduce binary variables  $\beta_p$ , and add the following set of constraints to set  $\beta_p = 1$  for all attacker-optimal non-interdicted plans:

$$\forall_p \quad \sum_{l \in L} U_a(l) s_l - \sum_{\alpha, t} C_a^\alpha y_{\alpha, t} \geq U_a(p) - Z \delta_p - \beta_p + \varepsilon, \quad (7.20a)$$

where  $\varepsilon \in (0, 1)$  is a sufficiently small number such that the requirement can only be satisfied by an attacker-optimal plan  $p$  if  $\beta_p = 1$ . Then, for each plan  $p$  with  $\beta_p = 1$ , the

following constraint will be effective, placing the desired limit on the defender's utility:

$$\forall_p \quad \sum_{l \in L} U_d(l) s_l \leq U_d(p) + Z(1 - \beta_p), \quad (7.21a)$$

with  $U_d(p)$  the (precomputed) defender utility of a plan  $p$ . Since Constraints 7.19 ensure that the goals reached using an attacker-optimal plan are represented by the variables  $s_l$ , Constraints 7.21, in combination with Constraints 7.20 will ensure that the worst plan is chosen for the defender of those that are optimal for the attacker *from the set of plans that have been generated thus far*.

The problem with the approach above is that since we only generate and add a single optimal plan for the attacker at any given time, rather than all optimal attacker plans, we may overestimate the worst-case utility for the defender. To handle this, once Algorithm 1 finds no additional constraints to add, we solve another integer program to check if the best response is, indeed, defender pessimal. To do this we need to modify the *IP* from Section 7.1 in two ways. First, we change the objective to minimize the defender's value (maximize her losses) for goals achieved by the attacker:

$$\min \quad \sum_l U_d(l) s_l$$

Second, we add a constraint that forces the attacker utility to equal the best response that we have computed in solving *DPIP-IP*, which we denote by  $\bar{U}$ :

$$\sum_l U_d(l) s_l - \sum_\alpha C_a^\alpha \sum_t y_{\alpha,t} = \bar{U}.$$

If the resulting objective value matches  $\sum_{l \in L} U_d(l) s_l$  computed by Constraints 7.21, we are done. If not, we have just identified a plan that can be added to the master program, at which point we restart Algorithm 1.

### 7.2.6 Heuristic Plan Interdiction

While we are the first to consider the problem of optimal plan interdiction at this level of generality, there are a number of informal ways in which attacker models, such as attack

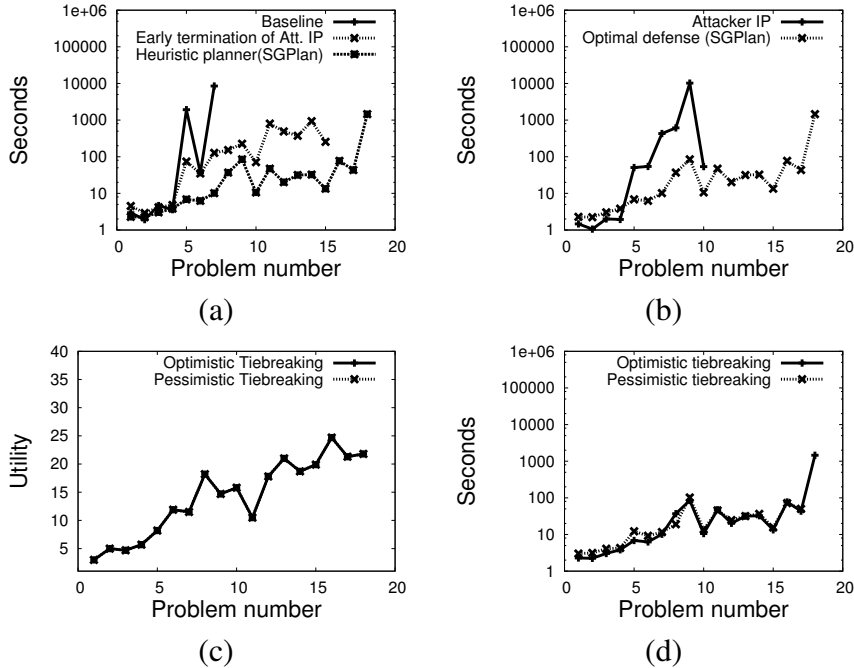


FIGURE 7.3: (a) Runtime comparison for several variants of constraint generation. (b) DPIP runtime vs. optimal planning runtime. (c) Runtime and (d) utility comparison of tie breaking approaches.

graphs or attack plans, are used to guide the design of mitigations in practice. In most cases, these actually require the specification of the entire attack graph, making it both suboptimal and intractable for even a modest number of state literals (for example, one would attempt to interdict a shortest path to a goal).

In this section, we propose a greedy heuristic, shown as Algorithm 2, which relies only on the ability to generate plans in response to specific mitigations. Our heuristic can be viewed as a kind of formalization of the way attack models are actually used to obtain mitigation strategies: that is, by iteratively interdicting attacker’s goals. Since different goals have different value to the defender, but may also carry varying interdiction costs, in each iteration we choose a goal to interdict to maximize the defender’s marginal utility. For simplicity, we assume that each mitigation blocks exactly one attack action, so that  $C_d^m = C_d^\alpha$  for action  $\alpha$  interdicted by mitigation  $m$ . In Algorithm 2, the function *chooseCheapest*( $p, g$ ) chooses the cheapest set of actions from plan  $p$  such that goal  $g$  is

---

**Algorithm 2:** Heuristic plan interdiction.

---

```
 $A_{cur} = A;$ 
while true do
   $(p, U, G(p)) = \text{optimalPlan}(A_{cur});$ 
   $U'_{best} = U;$ 
   $\tilde{A}_{best} = \emptyset;$ 
  for  $g \in G(p)$  do
     $\tilde{A} = \text{chooseCheapest}(p, g);$ 
     $(p', U', G(p')) = \text{optimalPlan}(A_{cur} \setminus \tilde{A});$ 
    if  $U' > U'_{best}$  then
       $U'_{best} = U';$ 
       $\tilde{A}_{best} = \tilde{A};$ 
    end
  end
  if  $U'_{best} - U > \sum_{\alpha \in \tilde{A}_{best}} C_a^D$  then
     $A_{cur} = A_{cur} \setminus \tilde{A}_{best};$ 
  else
    break;
  end
end
```

---

no longer satisfied, and  $G(p)$  represents the set of goals satisfied by a plan  $p$ .

Notice that since Algorithm 2 still requires us to compute optimal attacker plans, it is not a given that it will be faster than *DPIP\_IP*. Below, we compare *DPIP\_IP* with the greedy heuristic, demonstrating that in some cases it can be quite effective.

### 7.3 Experiments with deterministic plan interdiction

For the experimental evaluation, we used the *pathways* planning domain from the 2006 international planning competition (IPC). To formulate these as interdiction problems, we let the set of mitigations correspond to the set of plan actions (thus, each mitigation  $m$  blocks exactly one attack action). We let the defender losses equal attacker gains, that is,  $U_d(l) = -U_a(l)$  for all  $l \in L$ ; note that the game remains non-zero-sum, since the costs are accounted for differently by the defender and the attacker. Interdiction costs are fixed

at 1 for interdicting all actions  $\alpha \in A$ , with the exception of special actions generated to capture logical expressions or combinations of goals, for which the interdiction costs are set to infinity. The problems that we ran our experiments on ranged in size from 46 possible attacker actions and one potential goal (problem number 1) to 490 attacker actions and 27 potential goals (problem number 20), with a general trend of a larger index having a larger number of possible attacker actions and goals. All computational experiments were performed on a 64 bit Linux 2.6.18-164.el5 computer with 96 GB of RAM and two quad-core hyperthreaded Intel Xeon 2.93 GHz processors, unless otherwise specified. We did not make use of any parallel or multi-threading capabilities, restricting a solver to a single thread, when relevant. Integer programs were solved using CPLEX version 12.4.

In the first set of experiments we compare the performance of our proposed algorithms. (Note that comparing to alternatives such as DOBSS [47] is a non-starter, since there are vastly more feasible plans than could be stored in memory.) The results are shown in Figure 7.3 (a). The “Baseline” solution in the figure uses Algorithm 1 as is, that is, with constraints generated by solving the integer program in Section 7.1. As we can see, even with constraint generation, this approach scales rather poorly: indeed, on most problems it is entirely infeasible. If we terminate the planning *IP* early, the performance is now far more reasonable, and most IPC problems can be solved. Finally, using SGPLAN to generate constraints yields another improvement (up to a factor of 100 on some problem instances); indeed, the running time of *DPIP* is now 10-100 seconds on all but one problem.

The second set of experiments compares the solution time of *DPIP* to the time it takes to run the attack planning *IP* on the same problem instance. The result, shown in Figure 7.3 (b), is quite surprising: *DPIP* with constraint generation appears to be many orders of magnitude *easier* to solve, even though it uses the same structure as the attack planning *IP*! The reason is that even in the absence of generated plans, the master *IP* still ensures that some *feasible* plan is chosen by the attacker. The result is that the mitigations chosen

in the initial iteration already significantly restrict the planning problem, making it much easier to solve.

Our third set of experiments compares optimistic and pessimistic tie breaking. As one would expect, there is no discernible difference in terms of defender utility (Figure 7.3 (c)). What may be slightly more surprising is that there is essentially no difference in running time either (Figure 7.3 (d)).

Our fourth set of experiments compares the greedy heuristic for *DPIP* to our optimal plan interdiction algorithm (Figure 7.4). The results are mixed. On the one hand, the

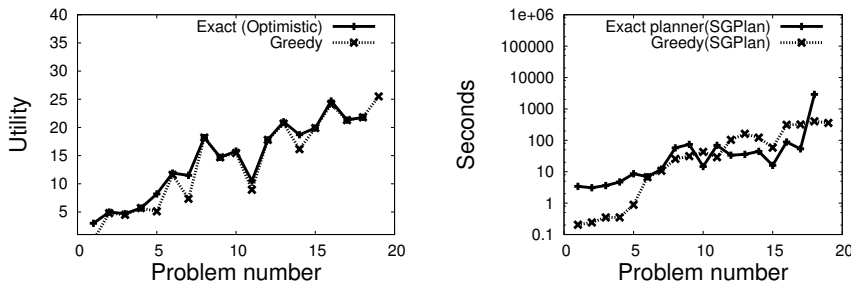


FIGURE 7.4: Comparison of the greedy heuristic (Algorithm 2) to optimal in terms of defender utility (left) and runtime (right).

heuristic is remarkably good at approximating the optimal interdiction strategy for the defender (Figure 7.4, left). Moreover, for the smaller (lower numbered) problems, it is an order of magnitude faster than Algorithm 1. On the other hand, for many of the larger problems (except the largest), the optimal interdiction algorithm is actually much faster than the heuristic (Figure 7.4, right). Finally, while the runtime comparison between the greedy heuristic and optimal *DPIP* is ambiguous, greedy wins hands down in terms of memory utilization: the memory footprint of greedy is about 10% that for Algorithm 1 on the larger problems.

## 7.4 Dealing with uncertainty

A crucial limiting feature of the approach we presented thus far is that it assumes determinism in every facet of the plan interdiction problem. In this section we incrementally relax this assumption.

### 7.4.1 Uncertainty about Attacker's Capabilities, Costs, and Goals

It is, in general, a severe simplification to treat attacker's capabilities, goals, and action costs as known to the defender. We relax this assumption in a relatively standard way by using a Bayesian Stackelberg game model, and focus here on uncertainty in attacker's capabilities (uncertainty about goals and costs can be handled similarly). Let  $\Theta$  be the set of attacker types, and let  $I_\theta$  be the set of capabilities (i.e., initial state) of type  $\theta \in \Theta$ . Additionally, let  $p_\theta$  be the probability that the attacker has type  $\theta$ , and index the variables  $s, U_a, U_d$ , and  $\delta$  by the attacker's type. The *DPIP* objective function becomes

$$\sum_{\theta} \sum_{l \in L} p_\theta U_d^\theta(l) s_{\theta,l} - \sum_{m \in M} D_m C_d^m,$$

while Constraints 7.16-7.19 now have to hold for each attacker type  $\theta$ . Finally, in the constraint generation algorithm we solve a separate planning problem for each type, and add all the corresponding plans to the master program.

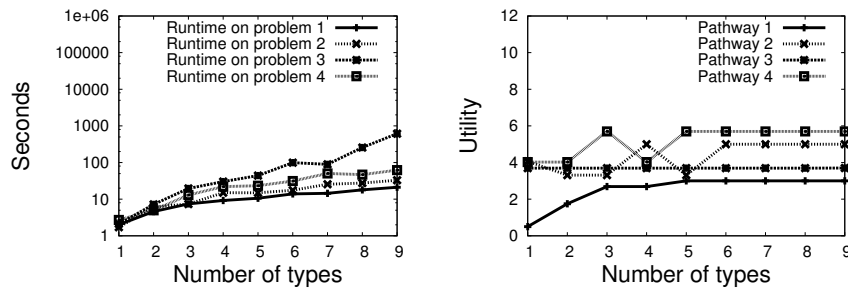


FIGURE 7.5: Runtime (left) and defender utility (right) of Bayesian plan interdiction as a function of the number of attacker types.

In Figure 7.5 (left) we show the runtime of the Bayesian plan interdiction framework

with the appropriate modifications to the master problem and constraint generation. For this figure, we first generated 9 attacker types corresponding to initial capabilities, and then incrementally abstracted these to obtain smaller numbers of types. The good news is that on the three of four problems the runtime trend is subexponential. Another piece of good news is that it appears that type abstraction is rather effective, and the utility of the defender reaches near optimal with the number of abstracted types less than half of the number of actual types (indeed, grouping all types into one seems to already yield a near-optimal solution on three of the four problems). The bad news is that the bottleneck of this approach seems to be memory (we used a machine with 8GB of memory for this set of experiments): we were unable to run larger problem instances due to memory limitations.

#### 7.4.2 *Uncertain Attack Execution with Retry*

Aside from Bayesian plan interdiction we had just considered, there is another type of uncertainty that we can handle within the basic *DPIP* framework: uncertainty about execution of attack actions, when the attacker can retry an action that fails an arbitrary number of times. The reason is that if the action cost is  $C_a^\alpha$ , and the probability that the action succeeds is  $P_\alpha$ , we can reformulate the problem by assigning a new action cost  $\bar{C}_a^\alpha = C_a^\alpha / P_\alpha$ , and then solve the corresponding *DPIP*.

#### 7.4.3 *MDP Interdiction Problem*

While certain special cases of uncertainty can be handled within the basic *DPIP* framework we introduced, in general we must move beyond it. In this section we explore one such generalization, modeling the attacker’s problem as a *Terminating Markov Control Problem (TMCP)*. A TMCP is quite similar to a MDP, but in every state  $\sigma$  there is a “cash out” action that yields a reward based on which goals are satisfied and moves the MDP into an absorbing state, and, in addition, every action in each state has a non-zero probability of transitioning into the absorbing state. The convenience of using such termination

probabilities (which can represent the probability of the attacker being caught) is that we can equivalently formulate the problem as an MDP with a discount factor that depends on current and next state,  $\gamma_{\sigma,\sigma'}$ . Let  $U_a^\sigma(\alpha)$  be the attacker's value (reward) in state  $\sigma$  if he takes action  $\alpha$ , and let  $I$  and  $G$  be the initial state, and the state in which all of the attacker's goals are satisfied, respectively. Finally, define  $T_{\sigma'}^{\alpha,\sigma}$  to be the probability of moving to state  $\sigma'$  from  $\sigma$  if the attacker takes action  $\alpha$ . We formulate this as an interdiction problem in precisely the same way as we had done for *DPIP*, endowing the defender with the set of mitigation strategies and associated costs, and assigning the defender a utility function  $U_d^\sigma(\alpha)$  for each state  $\sigma$  and attack action  $\alpha$  performed in that state.

As the first step we leverage a linear programming formulation for computing the optimal policy of a TMCP [19]:

$$\max_{y_{\alpha,\sigma}} \sum_{\alpha,\sigma} U_a^\sigma(\alpha) y_{\alpha,\sigma} \quad (7.22)$$

s.t. :

$$\forall_{\sigma' \neq I} \sum_{\sigma,a} (\delta_{\sigma,\sigma'} - T_{\sigma'}^{\alpha,\sigma}) y_{\alpha,\sigma} = 0 \quad (7.23)$$

$$\sum_{\alpha,\sigma} (\gamma_{\sigma,I} - T_I^{\alpha,\sigma}) y_{\alpha,\sigma} = 1 \quad (7.24)$$

$$\sum_{\alpha,\sigma} (\gamma_{\sigma,G} - T_G^{\alpha,\sigma}) y_{\alpha,\sigma} = -1, \quad (7.25)$$

where  $y_{\alpha,\sigma} = 1$  iff the attacker chooses action  $\alpha$  in state  $\sigma$ . The full MDP interdiction problem (*MDPIP*) can then be formulated as the following integer program:

$$\max \sum_{\alpha,\sigma} U_d^\sigma(\alpha) y_{\alpha,\sigma} - \sum_m C_d^m D_m$$

s.t. :

$$\forall_p \sum_{\alpha,\sigma} U_a^\sigma(\alpha) y_{\alpha,\sigma} \geq U_a(p) - \delta_p Z$$

constraints 7.14 – 7.18, 7.23 – 7.25.

We can then use Algorithm 1 as is, with constraints generated using standard methods for solving MDPs.

To study the difference between the runtime and memory characteristics of *MDPIP* and *DPIP*, we use the planning problem in Example 5, varying the number of actions available to the attacker between 13 and 16 (*MDPIP* runs out of memory for larger problems, as well as for all problems from IPC 2006). To introduce uncertainty, we added attack step execution uncertainty with retry; as we noted in Section 7.4.2, such uncertainty can also be captured in the *DPIP* framework. Figure 7.6 shows what we would have expected:

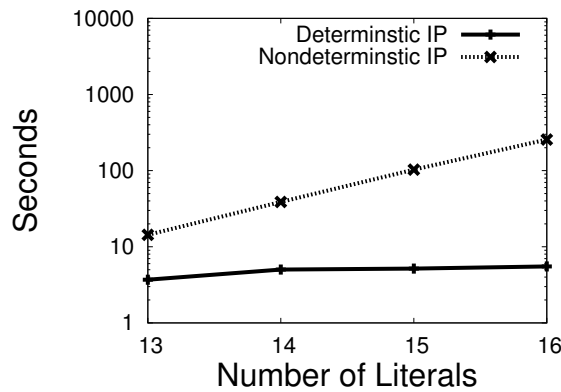


FIGURE 7.6: Runtime comparison between *MDPIP* and *DPIP*.

*MDPIP*, while much more general, is an order of magnitude (and growing) slower than *DPIP*. Additionally, the memory footprint of *MDPIP* is an even greater issue: for 16 literals, solving *MDPIP* takes about a factor of 20 more memory.

## Value of Commitment

In this chapter, we define and study the “value of commitment,” which is the ratio between player 1’s utility when she commits, and her utility when she does not commit. Perhaps the most closely related paper to our work in this chapter is “On the Value of Correlation” by Ashlagi et al. (2008). In this work, the authors consider the phenomenon that in some games, there are correlated equilibria that lead to higher welfare than any Nash equilibrium, and they study the ratio between the welfare in the best correlated equilibrium and the best Nash equilibrium. They call this ratio the *value of mediation*. This is similar to our notion of value of commitment, except that we consider player 1 (the leader)’s utility rather than the combined welfare of all the players, and are interested in the ratio between the solution with and without commitment (according to several solution concepts) rather than between correlated and Nash equilibrium. (Ashlagi et al. also study the ratio between the maximum possible welfare and the maximum welfare obtained in a correlated equilibrium.) This work by Ashlagi et al., in turn, builds upon existing work that tries to evaluate the welfare obtained under certain solution concepts using certain ratios.

Most notably, there is the work on the *price of anarchy* [31, 46]. The price of anarchy is a measure of the amount of welfare lost due to the selfishness of the players (relative

to what a central planner could obtain). Specifically, it looks at the worst possible ratio between the worst Nash equilibrium and the optimal social welfare obtainable from the game. Of particular interest is the cost of selfish routing [52], which considers the ratio between the latency in Nash equilibrium and the optimal latency. There is also a literature on *Stackelberg routing* [51, 26]. While the use of the word “Stackelberg” here indicates a superficial similarity to our work (in particular the routing games that we study later in the chapter), there is a fundamental difference. In the existing Stackelberg routing literature, there is a benevolent central authority that first routes some of the flow before the selfish players move, and the goal of the central authority is to minimize the resulting price of anarchy (i.e., the ratio between the cost of the resulting flow and the optimal flow). In contrast, in our setting, player 1 is not a benevolent central authority, but rather a selfish player like any other, who is trying to minimize her own cost (or maximize her own utility), as is usually the case in Stackelberg games; and we are interested in the improvement in her own cost/utility that the player can obtain through commitment.

Various other, similar ratios have been studied, such as the price of stability (which considers the best rather than the worst Nash equilibrium) [3], and analogous concepts using correlated rather than Nash equilibria [11]. However, to our knowledge, none of these ratios have considered the benefit of commitment.

In this chapter we first formally define the  $VoPC$ ,  $VoMC$ , and  $MvP$  ratios, in Section 8.1. In Section 8.2, as a “warm-up,” we investigate these ratios in normal-form and symmetric normal-form games. We investigate these ratios in extensive-form and security games in Sections 8.3 and 8.4. Finally, in Sections 8.5, 8.6, and 8.7, we investigate these ratios in atomic selfish routing games with linear, quadratic,  $k$ -nomial and arbitrary costs. This chapter is based on Letchford et al. [40]. We also evaluate these concepts experimentally over various distributions in a number of common game types in this paper.

## 8.1 A formal definition of value

In the literature on commitment, an advantage to committing to a strategy is often shown; moreover, we know that if commitment to a *mixed* strategy is possible, then at least in two-player games, committing to a mixed strategy never hurts [64]. However, to our knowledge, there has been no analysis of *how much* commitment can benefit a player. In this paper, we define and study the “value of commitment,” which is the ratio between player 1’s utility when she commits, and her utility when she does not commit. In the normal form game in Example 1.3, we considered two possibilities: player 1 can commit to a pure strategy, or to a mixed strategy. Without commitment, player 1 gets 1. With pure-strategy commitment, player 1 gets 2. Hence, the value of pure commitment (*VoPC*) for this game is  $2/1 = 2$ . With mixed-strategy commitment, player 1 gets 2.5. Hence, the value of mixed commitment (*VoMC*) for this game is  $2.5/1 = 2.5$ . We will also be interested in how much player 1 gains by committing to a mixed strategy rather than a pure one—this mixed vs. pure (*MvP*) ratio is  $2.5/2 = 1.25$  for this game. In fact, we will generally be interested in *classes* of games, and the *largest* values that these ratios can attain.

We now define the concepts of *VoPC*, *VoMC*, and *MvP* formally. The main difficulty in doing so is that we must specify which outcome materializes after player 1 commits, or when player 1 does not commit. That is, what solution concept is used for the remaining game? In Example 1.3, this was unambiguous: if player 1 commits, player 2 faces a straightforward single-agent optimization problem; if player 1 does not commit, the game is solvable by iterated strict dominance. However, we will not be so lucky for every game: the game may not be solvable by iterated strict dominance, and if there are at least three players then even after player 1’s commitment the game may not be solvable by iterated strict dominance. (When there are more than two players we assume that only player 1 is able to make a commitment.) Generally, we will have to use a solution concept that

is defined everywhere. For example, we can assume that the worst Nash equilibrium for player 1 will result—or the best, or the worst correlated equilibrium, etc.

Let  $SC$  denote a solution concept, and let  $u_1(SC(G))$  be the utility that player 1 gets under that solution concept in game  $G$ . Let  $S_1$  (resp.  $\Sigma_1$ ) be the set of player 1's pure (resp. mixed) strategies. (Of course,  $S_1 \subseteq \Sigma_1$ .) Let  $\sigma_1 \in \Sigma_1$  be a strategy for player 1; let  $G|_{\sigma_1}$  be the game among the remaining players that results after player 1 commits to  $\sigma_1$ . Generally we will be interested not in a single game  $G$ , but in a class  $\mathcal{G}$ . Then the value of pure commitment (w.r.t.  $SC$ ) is defined as  $VoPC^{SC}(\mathcal{G})$ :

$$VoPC^{SC}(\mathcal{G}) = \sup_{G \in \mathcal{G}} \frac{\sup_{s_1 \in S_1} \{u_1(SC(G|_{s_1}))\}}{u_1(SC(G))}$$

Similarly, we get the following definitions for the value of mixed commitment,  $VoMC^{SC}(\mathcal{G})$ :

$$VoMC^{SC}(\mathcal{G}) = \sup_{G \in \mathcal{G}} \frac{\sup_{\sigma_1 \in \Sigma_1} \{u_1(SC(G|_{\sigma_1}))\}}{u_1(SC(G))}$$

and the mixed vs. pure ratio,  $MvP^{SC}(\mathcal{G})$ :

$$MvP^{SC}(\mathcal{G}) = \sup_{G \in \mathcal{G}} \frac{\sup_{\sigma_1 \in \Sigma_1} \{u_1(SC(G|_{\sigma_1}))\}}{\sup_{s_1 \in S_1} \{u_1(SC(G|_{s_1}))\}}$$

Some games are more naturally modeled by cost functions  $c_i$  rather than utility functions  $u_i$ . In that case, we redefine

$$VoPC^{SC}(\mathcal{G}) = \sup_{G \in \mathcal{G}} \frac{c_1(SC(G))}{\inf_{s_1 \in S_1} \{c_1(SC(G|_{s_1}))\}}$$

etc. We are assuming nonnegative utility and cost functions everywhere.

We have the following trivial lemma:

**Lemma 1.** *For any  $SC$ ,  $VoMC^{SC}(\mathcal{G}) \geq VoPC^{SC}(\mathcal{G})$ .*

*Proof.* The sup in the numerator of VoMC is over a larger set than the sup in the numerator of VoPC (in the case of costs, the inf in the denominator of VoMC is over a larger set).  $\square$

For a solution concept such as iterated strict dominance (ISD), the expressions above are not well-defined because for games  $G$  that are not solvable by ISD,  $u_1(SC(G))$  is not defined. Hence, when we consider  $VoPC^{ISD}$  (etc.), we require that the outer sups are taken only over games solvable by ISD and the inner sups are taken only over strategies that result in a game solvable by ISD. Similarly, we can consider the solution concept UniqueNash, which can be applied only when the game has a unique Nash equilibrium. Because a game solvable by ISD has a unique correlated equilibrium, we get:

**Lemma 2.** *For any class of games  $\mathcal{C}$ , for any  $SC \in \{\text{BestNash}, \text{WorstNash}, \text{UniqueNash}, \text{BestCorrelated}, \text{WorstCorrelated}\}$ , we have  $VoPC^{ISD}(\mathcal{C}) \leq VoPC^{SC}(\mathcal{C})$ ,  $VoMC^{ISD}(\mathcal{C}) \leq VoMC^{SC}(\mathcal{C})$ , and  $MvP^{ISD}(\mathcal{C}) \leq MvP^{SC}(\mathcal{C})$ .*

As a result, if we can prove a high ratio for ISD, it immediately implies a high ratio for these other concepts. Similarly for UniqueNash, we get:

**Lemma 3.** *For any class of games  $\mathcal{C}$ , for any  $SC \in \{\text{BestNash}, \text{WorstNash}\}$ , we have  $VoPC^{\text{UniqueNash}}(\mathcal{C}) \leq VoPC^{SC}(\mathcal{C})$ ,  $VoMC^{\text{UniqueNash}}(\mathcal{C}) \leq VoMC^{SC}(\mathcal{C})$ , and  $MvP^{\text{UniqueNash}}(\mathcal{C}) \leq MvP^{SC}(\mathcal{C})$ .*

## 8.2 Normal-form and symmetric normal-form games

Let us start with some classes of games for which the ratios are easy to determine.

**Normal-form games.** First, we consider the class  $NF_{2 \times 2}$  of  $2 \times 2$  normal-form games.

**Theorem 28.**  $VoPC^{ISD}(NF_{2 \times 2}) = VoMC^{ISD}(NF_{2 \times 2}) = MvP^{ISD}(NF_{2 \times 2}) = \infty$ .

*Proof.* To prove  $VoPC^{ISD}(NF_{2 \times 2}) = \infty$ , consider the game in Figure 8.1. In this game,  $U$  strictly dominates  $D$ , so that the game is solvable by iterated strict dominance, resulting

in a utility of  $\varepsilon$  for player 1. If player 1 commits to the pure strategy  $D$ , player 2's best response will be  $R$ , resulting in a utility of  $1 - \varepsilon$  for player 1. Thus  $VoPC^{ISD}(NF_{2 \times 2}) \geq \frac{1-\varepsilon}{\varepsilon}$  for any  $\varepsilon$ , hence by letting  $\varepsilon \rightarrow 0$ , we get  $VoPC^{ISD}(NF_{2 \times 2}) = \infty$ .  $VoMC^{ISD}(NF_{2 \times 2}) = \infty$

	$L$	$R$
$U$	$(\varepsilon, 1)$	$(1, 0)$
$D$	$(0, 0)$	$(1 - \varepsilon, 1)$

FIGURE 8.1: Game for  $VoPC^{ISD}(NF_{2 \times 2})$  and  $VoPC^{ISD}(NF_{2 \times 2})$

	$L$	$R$
$U$	$(\varepsilon, 1)$	$(1, 0)$
$D$	$(0, 0)$	$(2\varepsilon, 1)$

FIGURE 8.2: Game for  $MvP^{ISD}(NF_{2 \times 2})$

follows from  $VoPC^{ISD}(NF_{2 \times 2}) = \infty$  and Lemma 1.

To prove  $MvP^{ISD}(NF_{2 \times 2}) = \infty$ , consider the game in Figure 8.2.

The optimal pure strategy to commit to is again  $D$ , but this time this only gives player 1 a utility of  $2\varepsilon$ . However, if player 1 can commit to a mixed strategy, then she can commit to  $\frac{1}{2} - \delta U, \frac{1}{2} + \delta D$ . Against this mixed strategy, player 2 still prefers to play  $R$ , and this way player 1 can get an expected utility arbitrarily close to  $1/2 + \varepsilon$ . Thus  $MvP^{ISD}(NF_{2 \times 2}) \geq (1/2 + \varepsilon)/(2\varepsilon)$ , hence by letting  $\varepsilon \rightarrow 0$ , we get  $MvP^{ISD}(NF_{2 \times 2}) = \infty$ .  $\square$

**Symmetric normal-form games.** Next, we consider the class  $SNF_{3 \times 3}$  of  $3 \times 3$  symmetric normal-form games.

**Theorem 29.**  $VoPC^{ISD}(SNF_{3 \times 3}) = VoMC^{ISD}(SNF_{3 \times 3}) = MvP^{ISD}(SNF_{3 \times 3}) = \infty$ .

*Proof.* To prove  $VoPC^{ISD}(SNF_{3 \times 3}) = \infty$ , consider the game in Figure 8.3. First,  $M$  and  $C$  are dominated by  $T$  and  $L$ , respectively. After removing these,  $B$  and  $R$  are then dominated by  $T$  and  $L$ , respectively, so that the game is solvable by ISD. This results in a utility of  $\varepsilon$  for player 1. If player 1 commits to the pure strategy  $M$ , player 2's best response will be  $R$ , resulting in a utility of  $1 - 3\varepsilon$  for player 1. Thus,  $VoPC^{ISD}(SNF_{3 \times 3}) \geq \frac{1-4\varepsilon}{\varepsilon}$  for any  $\varepsilon$ ,

hence by letting  $\varepsilon \rightarrow 0$ , we get  $VoPC^{ISD}(SNF_{3 \times 3}) = \infty$ . Again,  $VoMC^{ISD}(SNF_{3 \times 3}) = \infty$  follows from  $VoPC^{ISD}(SNF_{3 \times 3}) = \infty$  and Lemma 1.

	$L$	$C$	$R$
$U$	$(\varepsilon, \varepsilon)$	$(1-2\varepsilon, 0)$	$(1, 0)$
$M$	$(0, 1-2\varepsilon)$	$(1-3\varepsilon, 1-3\varepsilon)$	$(1-4\varepsilon, 1-\varepsilon)$
$D$	$(0, 1)$	$(1-\varepsilon, 1-4\varepsilon)$	$(0, 0)$

FIGURE 8.3: Game for  $VoPC^{ISD}(SNF_{3 \times 3})$  and  $VoMC^{ISD}(SNF_{3 \times 3})$

To prove  $VoMC^{ISD}(SNF_{3 \times 3}) = \infty$ , consider the game in Figure 8.4. The optimal pure strategy to commit to is again  $M$ , but this time this only gives player 1 a utility of  $2\varepsilon$ . However, if player 1 can commit to a mixed strategy, then she can commit to  $\frac{1}{2} T, \frac{1}{2} M$ . Against this mixed strategy, player 2 still prefers to play  $R$ , and this way player 1 gets an expected utility of  $1/2 + \varepsilon$ . Thus  $MvP^{ISD}(SNF_{3 \times 3}) \geq (1/2 + \varepsilon)/(2\varepsilon)$  for any  $\varepsilon$ , hence by letting  $\varepsilon \rightarrow 0$ , we get  $MvP^{ISD}(SNF_{3 \times 3}) = \infty$ .  $\square$

	$L$	$C$	$R$
$U$	$(\varepsilon, \varepsilon)$	$(\frac{1}{2} + \varepsilon, 0)$	$(1, 0)$
$M$	$(0, \frac{1}{2} + \varepsilon)$	$(\frac{1}{2}, \frac{1}{2})$	$(2\varepsilon, 1-\varepsilon)$
$D$	$(0, 1)$	$(1-\varepsilon, 2\varepsilon)$	$(0, 0)$

FIGURE 8.4: Game for  $MvP^{ISD}(SNF_{3 \times 3})$

### 8.3 Extensive-form games

Next we will consider the class  $PIEF$  of perfect-information extensive-form games.

**Theorem 30.**  $VoPC^{SPNE}(PIEF) = VoMC^{SPNE}(PIEF) = MvP^{SPNE}(PIEF) = \infty$ .

*Proof.* To prove  $VoPC^{SPNE}(PIEF) = \infty$ , consider the game in Figure 8.5. (It should be emphasized that player 1, the player with commitment power, moves *second* in this game.) In this game,  $L$  dominates  $R$  for player 1. Given this, at the top level  $L$  dominates  $R$  for player 2, resulting in a utility of  $\varepsilon$  for player 1. If player 1 commits to  $R$ , then player 2 will prefer to go right, resulting in a utility of  $1 - \varepsilon$  for player 1. Thus  $VoPC^{SPNE}(PIEF) \geq \frac{1-\varepsilon}{\varepsilon}$ , hence by letting  $\varepsilon \rightarrow 0$ , we get  $VoPC^{SPNE}(PIEF) = \infty$ . As usual,  $VoMC^{SPNE}(PIEF) = \infty$  follows from  $VoPC^{SPNE}(PIEF) = \infty$  and Lemma 1.

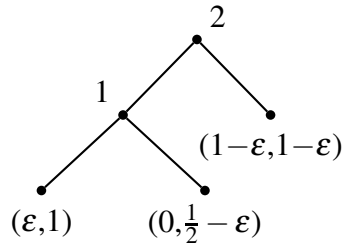


FIGURE 8.5: Game for  $VoPC^{SPNE}(PIEF)$  and  $VoMC^{SPNE}(PIEF)$

To prove  $MvP^{SPNE}(PIEF) = \infty$ , consider the game in Figure 8.6. The optimal strategy to commit to is  $L$ , leading to a utility of  $2\varepsilon$  for player 1. If player 1 commits to a mixed strategy of  $\frac{1}{2} R$ ,  $\frac{1}{2} L$ , then player 2 will still prefer to go right. In this case player 1's expected utility is  $\frac{1-\varepsilon}{2}$ . Thus  $MvP^{SPNE}(PIEF) \geq ((1 - \varepsilon)/2)/\varepsilon$ , hence by letting  $\varepsilon \rightarrow 0$ , we get  $MvP^{SPNE}(PIEF) = \infty$ .  $\square$

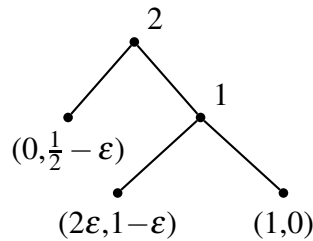


FIGURE 8.6: Game for  $MvP^{SPNE}(PIEF)$

## 8.4 Security games

Motivated by the various applications of computing optimal mixed strategies to commit to in security domains (as discussed in the introduction), we now consider a small subclass of the class of security games defined by Kiekintveld et al. (2009). We will call this class *simple security games* (SSG). In a simple security game, there is a set of *targets*  $T$ . The defender (player 1) chooses a target to defend; the attacker (player 2) chooses a target to attack. Each player's utility depends on two things: (1) which target the attacker attacks, and (2) whether the defender has chosen to defend that target. Thus,  $u_1^c(t)$  (resp.,  $u_2^c(t)$ ) is the defender's (resp., attacker's) utility when target  $t$  is attacked and it is defended, and  $u_1^u(t)$  (resp.,  $u_2^u(t)$ ) is the defender's (resp., attacker's) utility when target  $t$  is attacked and it is not defended. We require that for any  $t$ ,  $u_1^c(t) > u_1^u(t)$  and  $u_2^c(t) < u_2^u(t)$ .

**Theorem 31.**  $VoPC^{\text{UniqueNash}}(SSG_{2 \times 2}) = VoMC^{\text{UniqueNash}}(SSG_{2 \times 2}) = MvP^{\text{UniqueNash}}(SSG_{2 \times 2}) = \infty$ .

*Proof.* To prove  $VoPC^{\text{UniqueNash}}(SSG_{2 \times 2}) = \infty$ , consider the game in Figure 8.7. This game has a unique Nash equilibrium, consisting of the mixed strategy  $\frac{1}{2} T_1, \frac{1}{2} T_2$  for player 1 and the mixed strategy  $(1 - \frac{2\varepsilon}{1+2\varepsilon}) T_1$  for player 2,  $\frac{2\varepsilon}{1+2\varepsilon} T_2$ . This leads to an expected utility for player 1 of  $(\frac{3}{4})(\frac{2\varepsilon}{1+2\varepsilon}) + (\frac{\varepsilon}{2})(1 - \frac{2\varepsilon}{1+2\varepsilon}) < 2\varepsilon$ . If player 1 commits to the pure strategy  $T_1$ , player 2's best response is  $T_2$ , and player 1's utility is  $\frac{1}{2}$ . Thus  $VoPC^{\text{UniqueNash}}(SSG_{2 \times 2}) \geq (1/2)/(2\varepsilon)$ , hence by letting  $\varepsilon \rightarrow 0$ , we get  $VoPC^{\text{UniqueNash}}(SSG_{2 \times 2}) = \infty$ .  $VoMC^{\text{UniqueNash}}(SSG_{2 \times 2}) = \infty$  follows from  $VoPC^{\text{UniqueNash}}(SSG_{2 \times 2}) = \infty$  and Lemma 1.

To prove  $MvP^{\text{UniqueNash}}(SSG_{2 \times 2}) = \infty$ , consider the game in Figure 8.8. Again, the optimal pure strategy to commit to is  $T_1$ , but this time it only gives player 1 a utility of  $2\varepsilon$ . However, if player 1 can commit to a mixed strategy, then she can commit to  $\frac{1}{2} + \delta T_1, \frac{1}{2} - \delta T_2$  (for some small  $\delta > 0$ ). Against this mixed strategy, player 2

	$T_1$	$T_2$
$T_1$	$(\varepsilon, 0)$	$(\frac{1}{2}, 1)$
$T_2$	$(0, 1)$	$(1, 0)$

FIGURE 8.7: Game for  $VoPC^{\text{UniqueNash}}(SSG_{2 \times 2})$  and  $VoMC^{\text{UniqueNash}}(SSG_{2 \times 2})$

	$T_1$	$T_2$
$T_1$	$(\varepsilon, 0)$	$(2\varepsilon, 1)$
$T_2$	$(0, 1)$	$(1, 0)$

FIGURE 8.8: Game for  $MvP^{\text{UniqueNash}}(SSG_{2 \times 2})$

still prefers to play  $T_2$ , and thus player 1 can get an expected utility arbitrarily close to  $\frac{1}{2} + \varepsilon$ . Thus  $MvP^{\text{UniqueNash}}(SSG_{2 \times 2}) \geq (1/2 + \varepsilon)/(2\varepsilon)$ , hence by letting  $\varepsilon \rightarrow 0$ , we get  $MvP^{\text{UniqueNash}}(SSG_{2 \times 2}) = \infty$ .  $\square$

Korzhyk et al. (2011c) show that in security games (satisfying a minor assumption), the optimal mixed strategy to commit to for the defender is always a Nash equilibrium strategy as well. This may seem to contradict our result that commitment can be of value in these games, but in fact there is no contradiction. The optimal thing to do when committing to a mixed strategy is to commit to a strategy that is very close to the equilibrium strategy, but incentivizes the attacker to play a response that is much better for the defender than the attacker equilibrium strategy. Also, in that paper the attacker breaks ties in the defender's favor, so that the optimal mixed strategy to commit to is actually exactly the defender's equilibrium strategy.

## 8.5 Routing games

We now consider what is known as an *atomic selfish routing* game. We will first focus on the simplest case where each player controls an indivisible unit amount of traffic and for every edge, the cost of using that edge scales linearly with the number of players using that edge. Thus, each edge  $e$  has a coefficient  $c_e$ , and if  $n_e$  players use that edge, each of them incurs a cost of  $c_e n_e$ . We will refer to this class of games as *LAR* (Linear Atomic Routing). Note that these games are naturally modeled by cost functions rather than utilities, and

thus we will be using the corresponding versions of the definitions of  $VoPC$ ,  $VoMC$ , and  $MvP$ . Our main theorem is the following:

**Theorem 32.** *For any solution concept  $SC \in \{\text{ISD}, \text{BestNash}, \text{WorstNash}, \text{BestCorrelated}, \text{WorstCorrelated}\}$ , we have  $VoPC^{SC}(LAR_{n \text{ player}}) = VoMC^{SC}(LAR_{n \text{ player}}) = MvP^{SC}(LAR_{n \text{ player}}) = n$  (where  $n$  is the number of players).*

*Proof.* We first show that  $VoMC^{SC}(LAR_{n \text{ player}}) \leq n$  for any solution concept  $SC$  where player 1 best-responds to the other players and the other players do not play dominated strategies (which includes all the solution concepts in the statement of the theorem)—and hence, by Lemma 1,  $VoPC^{SC}(LAR_{n \text{ player}}) \leq n$ . This part is simple. For any given routing game  $G$ , we can remove all the players but player 1 to obtain a game  $G^1$ . In this game, player 1 will just choose the lowest-cost path. That same path is still an option in  $G$ ; in the worst case, each of the other  $n - 1$  players uses all the edges on that path (but they will use each edge at most once because using an edge more than once is a dominated strategy). Because player 1 best-responds under  $SC$ , she must do at least that well; hence, we have  $c_1(SC(G)) \leq nc_1(SC(G^1))$ . Moreover, no matter which strategy  $\sigma_1$  player 1 commits to, she can do no better than  $c_1(SC(G^1))$ ; hence,  $c_1(SC(G|\sigma_1)) \geq c_1(SC(G^1))$ . It follows that for any game  $nc_1(SC(G|\sigma_1)) \geq c_1(SC(G))$ , hence  $VoMC^{SC}(LAR_{n \text{ player}}) \leq n$ . (The preceding discussion assumes that  $SC$  can be used to solve  $G$  and  $G|\sigma_1$ , which may not be the case if, for example,  $SC = \text{ISD}$ , but this does not affect upper bounds on the ratio.) In fact, from this we can also conclude that  $MvP^{SC}(LAR_{n \text{ player}}) \leq n$ : under commitment to a pure strategy, at the very least player 1 can commit to the path  $s_1$  she would choose in  $G^1$ , so that  $c_1(SC(G|s_1)) \leq nc_1(SC(G^1))$ . Then, for any  $\sigma_1$ ,  $nc_1(SC(G|\sigma_1)) \geq nc_1(SC(G^1)) \geq c_1(SC(G|s_1))$ , so  $MvP^{SC}(LAR_{n \text{ player}}) \leq n$ .

We now prove the more difficult part:  $VoPC^{\text{ISD}}(LAR_{n \text{ player}}) \geq n$  (which immediately implies  $VoMC^{\text{ISD}}(LAR_{n \text{ player}}) \geq n$  by Lemma 1, and it implies the same bounds for the other solution concepts by Lemma 2). For ease of presentation, we first consider the case

$n = 2$  (single follower) and then  $n = 3, 4$ . Let us consider the 2-player *LAR* game in Figure 8.9. player 1 (the Stackelberg leader) wishes to route her unit of flow from  $S_1$  to  $T_1 = T$ , and player 2 wishes to route his unit of flow from  $S_2$  to  $T_2 = T$ . The cost of using an edge is the value associated with that edge, multiplied by the number of players using that edge. Next, we show that this game induces a  $VoPC^{ISD}$  of 2.

Without commitment, this routing game is solvable by ISD, as follows. First of all, it is easy to see that any strategy that visits the same vertex twice is dominated, so we can remove such strategies. After that, it is a dominant strategy for player 1 to take the path  $S_1 \rightarrow B \rightarrow T$ . Given this path for player 1, the best-response strategy for player 2 is to take the path  $S_2 \rightarrow A \rightarrow B \rightarrow T$ . This ISD solution gives player 1 a cost of  $2 + \epsilon$ .

When we consider pure strategy commitment, if player 1 commits to the path  $S_1 \rightarrow A \rightarrow B \rightarrow T$ , then player 2's best response to is to take the alternate path directly from  $S_2 \rightarrow T$ . This results in a cost of  $1 + 2\epsilon$  for player 1. It follows that  $VoPC^{ISD}(LAR_{2 \text{ player}}) \geq \frac{2+\epsilon}{1+2\epsilon}$ , hence by letting  $\epsilon \rightarrow 0$ , we get  $VoPC^{ISD}(LAR_{2 \text{ player}}) \geq 2$ .

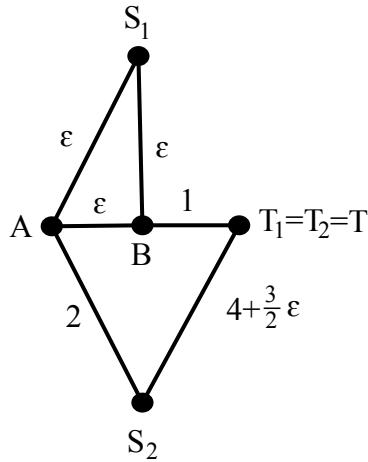


FIGURE 8.9: 2-player *LAR* routing game for  $VoPC$  and  $VoMC$

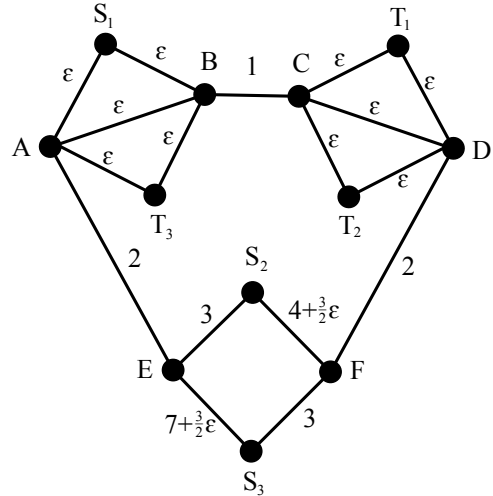


FIGURE 8.10: 3-player *LAR* routing game for  $VoPC$  and  $VoMC$

The three-player case for  $VoPC^{ISD}(LAR_{n \text{ player}})$  builds upon this same idea, namely that

the ISD solution is unable to dissuade the other players from using the central link, but the pure strategy commitment solution is able to do so at a minimal cost. Let us consider the game in Figure 8.10. Next, we show that this game induces a  $VoPC^{ISD}$  of 3.

Again, after removing the strategies that visit a node more than once, player 1 has a dominant strategy, namely  $S_1 \rightarrow B \rightarrow C \rightarrow T_1$ . Once we fix this strategy for player 1, player 3 has a dominant strategy, namely  $S_3 \rightarrow F \rightarrow D \rightarrow C \rightarrow B \rightarrow T_3$  (i.e., this is optimal regardless of what player 2 does). Once we fix this strategy, player 2 has a best response of  $S_2 \rightarrow E \rightarrow A \rightarrow B \rightarrow C \rightarrow T_2$ . This ISD solution gives player 1 a cost of  $3 + 2\epsilon$ .

When we consider pure strategy commitment, if player 1 commits to the path  $S_1 \rightarrow A \rightarrow B \rightarrow C \rightarrow D \rightarrow T_1$ , then player 2 has a dominant strategy  $S_2 \rightarrow F \rightarrow D \rightarrow T_2$  (i.e., this is optimal regardless of what player 3 does). Once we fix this strategy, player 3 has a best response of  $S_3 \rightarrow E \rightarrow A \rightarrow T_3$ . This results in a cost of  $1 + 4\epsilon$  for player 1. It follows that  $VoPC^{ISD}(LAR_3 \text{ player}) \geq \frac{3+2\epsilon}{1+4\epsilon}$ , hence by letting  $\epsilon \rightarrow 0$ , we get  $VoPC^{ISD}(LAR_3 \text{ player}) \geq 3$ .

The game for the four-player case for  $VoPC^{ISD}(LAR_n \text{ player})$  is pictured in Figure 8.11. Without commitment, the ISD solution is as follows: it is dominant for player 1 to choose  $S_1 \rightarrow B \rightarrow E \rightarrow T_1$ , then for player 4 to choose  $S_4 \rightarrow G \rightarrow C \rightarrow B \rightarrow E \rightarrow T_4$ , then for player 3 to choose  $S_3 \rightarrow J \rightarrow D \rightarrow E \rightarrow B \rightarrow T_3$ , and finally for player 2 to choose  $S_2 \rightarrow H \rightarrow A \rightarrow B \rightarrow E \rightarrow T_2$ . As a result, player 1's cost is  $4 + 2\epsilon$ . On the other hand, if player 1 can commit to a pure strategy, then she can commit to the path  $S_1 \rightarrow A \rightarrow B \rightarrow C \rightarrow S_1 \rightarrow B \rightarrow E \rightarrow T_1$ ; it then becomes dominant for player 2 to choose  $S_2 \rightarrow J \rightarrow D \rightarrow T_2$ , then for player 3 to choose  $S_3 \rightarrow G \rightarrow C \rightarrow T_3$ , and finally for player 4 to choose  $S_4 \rightarrow I \rightarrow F \rightarrow T_4$ . As a result, player 1's cost is  $1 + 4\epsilon$ . We can conclude that  $VoPC^{ISD}(LAR_4 \text{ player}) \geq 4$ .

The case of general  $n$  continues this pattern, the most important properties of which we describe now. In the previous constructions, we can break the graph up into two sections, an upper section focused around player 1's commitment choices, and a lower section setting up the two possible paths for each follower. When there is an even number of fol-

lowers, the bottom section is a closed loop (as in Figure 8.10), and when there is an odd number of followers, the bottom section is an open chain (as in Figure 8.11). The cost on the edges that connect the bottom and top sections must be at least  $n - 1$ . The top section is such that player 1 has a dominant strategy, which involves going through a single edge with coefficient 1 as well as two edges with coefficient  $\varepsilon$ . If we fix this path for player 1, then player  $n$  will have a dominant strategy that involves going through the same edge with coefficient 1; fixing that, the same becomes true for player  $n - 1$ , etc. This results in a cost of  $n + 2\varepsilon$  for player 1. However, the top section also allows player 1 to commit to a pure strategy that involves four edges with coefficient  $\varepsilon$ . If we fix this path for player 1, then player 2 will have a dominant strategy that avoids going through that edge with coefficient 1; fixing that, the same becomes true for player 3, etc. This results in a cost of  $1 + 4\varepsilon$  for player 1. It follows that  $VoPC^{ISD}(LAR_n \text{ player}) \geq \frac{n+2\varepsilon}{1+4\varepsilon}$ , hence by letting  $\varepsilon \rightarrow 0$ , we get  $VoPC^{ISD}(LAR_n \text{ player}) \geq n$ .

Again,  $VoMC^{ISD}(LAR_n \text{ player}) \geq n$  follows from  $VoPC^{ISD}(LAR_n \text{ player}) \geq n$  and Lemma 1.

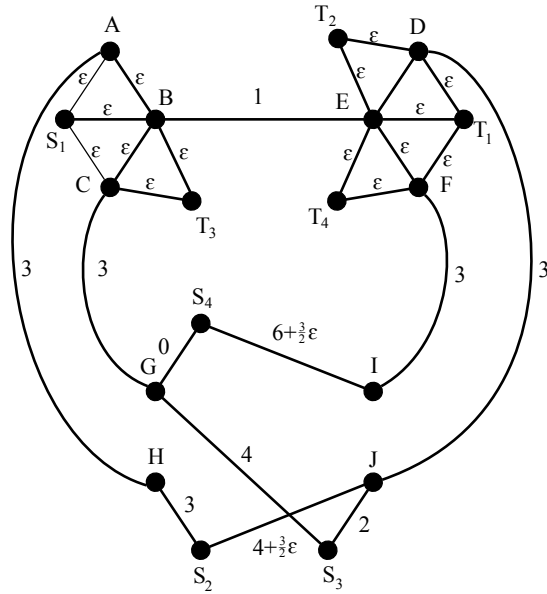


FIGURE 8.11: 4-player  $LAR$  game for  $VoPC$  and  $VoMC$

Finally, we need to show  $MvP^{ISD}(LAR_n \text{ player}) \geq n$  (which implies the same bound for

the other solution concepts by Lemma 2). The construction for  $MvP^{ISD}(LAR_n \text{ player})$  is similar to the one for  $VoPC^{ISD}(LAR_n \text{ player})$ ; to illustrate what needs to be modified, we highlight the differences with the three-player (two-follower) case. Let us consider the game in Figure 8.12. This game is identical to the one in Figure 8.10, except for some changes to the edge costs. The major change is in the upper section of the game, where all of the coefficients of  $\varepsilon$  have been replaced with coefficients of  $1/2$ , with the exception of the ones on the most direct path for player 1. Second, there are minor adjustments to the coefficients in the bottom section to compensate for the changes in the top section, ensuring similar comparisons as in the original version.

The effect of the changes is that commitment to a pure strategy hardly helps player 1 at all relative to taking the direct path (which results in a cost of  $3 + 2\varepsilon$ ). For example, if player 1 commits to the path  $S_1 \rightarrow A \rightarrow B \rightarrow C \rightarrow D \rightarrow T_1$ , then players 2 and 3 will not take the edge  $B \rightarrow C$ , but this still leaves player 1 with a cost of 3. She can also commit to the path  $S_1 \rightarrow A \rightarrow B \rightarrow C \rightarrow T_1$ , in which case player 2 will not take the edge  $B \rightarrow C$ , but player 3 will, so that player 1 ends up with a cost of  $3 + \varepsilon$ . In contrast, if player 1 can commit to a mixed strategy, she can randomize between using the path extended on the left side ( $S_1 \rightarrow A \rightarrow B \rightarrow C \rightarrow T_1$ )  $2\varepsilon$  of the time, the path extended on the right side ( $S_1 \rightarrow B \rightarrow C \rightarrow D \rightarrow T_1$ )  $2\varepsilon$  of the time, and the shortest path ( $S_1 \rightarrow B \rightarrow C \rightarrow T_1$ ) the rest of the time. This is enough to incentivize both of the followers to not use the edge between  $B$  and  $C$ , causing the expected cost for player 1 to be  $4\varepsilon(1 + \varepsilon) + (1 - 4\varepsilon)(1 + 2\varepsilon) = 1 + 2\varepsilon - 4\varepsilon^2$ . Letting  $\varepsilon \rightarrow 0$ , we obtain  $MvP^{ISD}(LAR_3 \text{ player}) \geq 3$ . This modification can in fact be applied to the  $VoPC^{ISD}(LAR_n \text{ player})$  construction for any  $n$ , giving us  $MvP^{ISD}(LAR_n \text{ player}) \geq n$ .

□

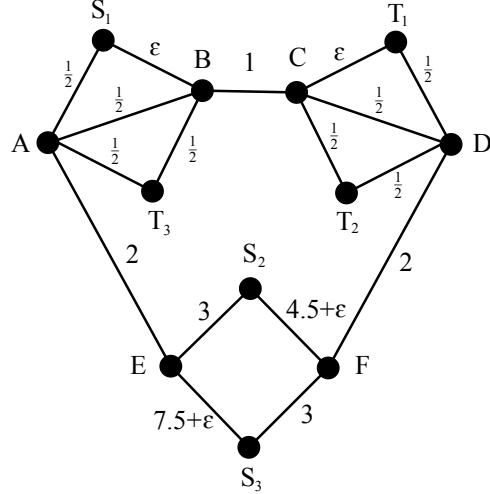


FIGURE 8.12: 3-player *LAR* game for *MvP*

## 8.6 Quadratic atomic routing games

For the sake of simplifying the presentation, we will first consider the case where the cost of using an edge scales quadratically with the number of players using the edge. Thus, each edge  $e$  has a coefficient  $c_e$ , and if  $n_e$  players use that edge, each of them incurs a cost of  $c_e n_e^2$ . We will refer to this class of games as *QAR* (Quadratic Atomic Routing).

**Theorem 33.** *For any solution concept  $SC \in \{\text{ISD}, \text{BestNash}, \text{WorstNash}, \text{BestCorrelated}, \text{WorstCorrelated}\}$ , we have  $\text{VoPC}^{SC}(\text{QAR}_{n \text{ player}}) = \text{VoMC}^{SC}(\text{QAR}_{n \text{ player}}) = \text{MvP}^{SC}(\text{QAR}_{n \text{ player}}) = n^2$  (where  $n$  is the number of players).*

*Proof.* We first show that  $\text{VoMC}^{SC}(\text{QAR}_{n \text{ player}}) \leq n^2$  for any solution concept  $SC$  where player 1 best-responds to the other players and the other players do not play dominated strategies (which includes all the solution concepts in the statement of the theorem)—and hence, by Lemma 1,  $\text{VoPC}^{SC}(\text{LAR}_{n \text{ player}}) \leq n^2$ . This part is simple. For any given routing game  $G$ , we can remove all the players but player 1 to obtain a game  $G^1$ . In this game, player 1 will just choose the lowest-cost path. That same path is still an option in  $G$ ; in the worst case, each of the other  $n - 1$  players uses all the edges on that path (but they will use each edge at most once because using an edge more than once is a dominated strategy).

Because player 1 best-responds under  $SC$ , she must do at least that well; hence, we have  $c_1(SC(G)) \leq n^2 c_1(SC(G^1))$ . Moreover, no matter which strategy  $\sigma_1$  player 1 commits to, she can do no better than  $c_1(SC(G^1))$ ; hence,  $c_1(SC(G|\sigma_1)) \geq c_1(SC(G^1))$ . It follows that for any game  $n^2 c_1(SC(G|\sigma_1)) \geq c_1(SC(G))$ , hence  $VoMC^{SC}(QAR_n \text{ player}) \leq n^2$ . (The preceding discussion assumes that  $SC$  can be used to solve  $G$  and  $G|\sigma_1$ , which may not be the case if, for example,  $SC = \text{ISD}$ , but this does not affect upper bounds on the ratio.) In fact, from this we can also conclude that  $MvP^{SC}(QAR_n \text{ player}) \leq n^2$ : under commitment to a pure strategy, at the very least player 1 can commit to the path  $s_1$  she would choose in  $G^1$ , so that  $c_1(SC(G|_{s_1})) \leq n^2 c_1(SC(G^1))$ . Then, for any  $\sigma_1$ ,  $n^2 c_1(SC(G|\sigma_1)) \geq n^2 c_1(SC(G^1)) \geq c_1(SC(G|_{s_1}))$ , so  $MvP^{SC}(QAR_n \text{ player}) \leq n^2$ .

We now prove what again turns out to be the more difficult part:  $VoPC^{\text{ISD}}(QAR_n \text{ player}) \geq n^2$ . The construction for this is very similar to the one in Theorem 32. Let us highlight the differences with the three-player (two-follower) case. Let us consider the game in Figure 8.13 (as compared to Figure 8.10). The major change in this game is in the coefficients on the edges between the upper and lower sections of the graph ( $A \rightarrow E$  and  $D \rightarrow F$ ). Instead of being  $n - 1$  ( $= 2$ ), they are now  $n^2 - 1$  ( $= 8$ ). This also necessitates some changes in the coefficients on the edges in the bottom section.

Let us now show that this game induces a  $VoPC^{\text{ISD}}(QAR_3 \text{ player}) \geq 9 = 3^2$ . Again, after removing the obviously dominated strategies that visit a node more than once, player 1 has a dominant strategy of  $S_1 \rightarrow B \rightarrow C \rightarrow T_1$ . Once we fix this strategy for player 1, player 3 has a dominant strategy, namely  $S_3 \rightarrow F \rightarrow D \rightarrow C \rightarrow B \rightarrow T_3$ . Once we fix this, player 2 has a best response of  $S_2 \rightarrow E \rightarrow A \rightarrow B \rightarrow C \rightarrow T_2$ . This ISD solution gives player 1 a cost of  $9 + 2\epsilon$ .

When we consider pure strategy commitment, if player 1 commits to the path  $S_1 \rightarrow A \rightarrow B \rightarrow C \rightarrow D \rightarrow T_1$ , then in the remaining game player 2 has a dominant strategy of  $S_2 \rightarrow F \rightarrow D \rightarrow T_2$ . This leads to player 3 having a best response of  $S_3 \rightarrow E \rightarrow A \rightarrow T_3$ , and results in a cost of  $1 + 4\epsilon$  for player 1. It follows that  $VoPC^{\text{ISD}}(QAR_3 \text{ player}) \geq \frac{9+2\epsilon}{1+4\epsilon}$

for any  $\varepsilon$ , hence by letting  $\varepsilon \rightarrow 0$ , we get  $VoPC^{ISD}(QAR_{3 \text{ player}}) \geq 9$ .

The general construction for  $VoPC^{ISD}(QAR_n \text{ player})$  follows exactly the same pattern as in Theorem 32, with the same changes as discribed above. The ISD solution in this game will have a cost of  $n^2 + 2\varepsilon$ , and the best pure-strategy commitment will result in a cost of  $1 + 4\varepsilon$  for player 1. Thus  $VoPC^{SC}(QAR_n \text{ player}) \geq \frac{n^2+2\varepsilon}{1+4\varepsilon}$  for any  $\varepsilon$ , hence by letting  $\varepsilon \rightarrow 0$ , we get  $VoPC^{ISD}(QAR_n \text{ player}) \geq n^2$ .  $VoMC^{ISD}(QAR_n \text{ player}) \geq n^2$  follows from  $VoPC^{ISD}(QAR_n \text{ player}) \geq n^2$  and Lemma 1.

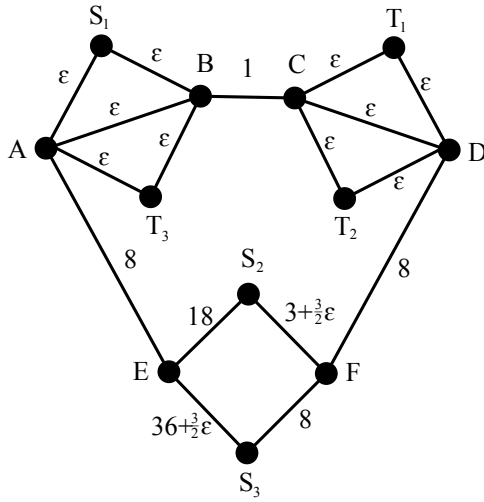


FIGURE 8.13: 3-player  $QAR$  game for  $VoPC$  and  $VoMC$

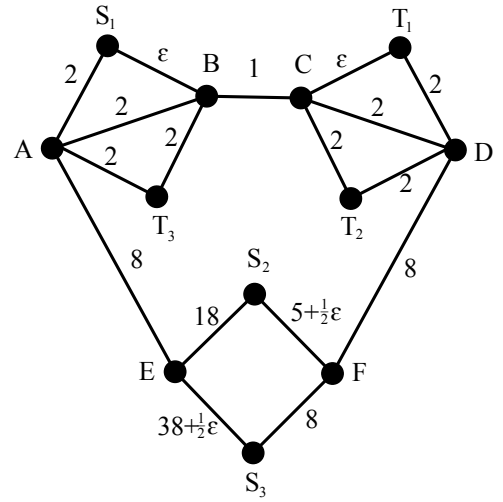


FIGURE 8.14: 3-player  $QAR$  game for  $MvP$

Let us now prove  $MvP^{ISD}(QAR_n \text{ player}) \geq n^2$ . The construction for  $MvP^{ISD}(QAR_n \text{ player})$  combines elements of the construction for  $MvP^{ISD}(LAR_n \text{ player})$  and the construction for  $VoPC^{SC}(QAR_n \text{ player})$ . As in Theorem 32, we take the construction for  $VoPC^{SC}(QAR_n \text{ player})$  and make a few changes to it. Namely, we increase the edge coefficients in the upper section of the graph (except those on the shortest (ISD) path  $S_1 \rightarrow B \rightarrow C \rightarrow T_1$ ) from  $\varepsilon$  to  $\frac{n^2-1}{4}$ . To compensate for this, we also need to make some changes to the coefficients in the bottom section. We illustrate this in Figure 8.14 for the 3-player case.

Again, this construction causes commitment to a pure strategy to hardly help player 1

at all. player 1's best pure strategy commitment is still the path  $S_1 \rightarrow A \rightarrow B \rightarrow C \rightarrow DT_1$ , which will be enough to incentivize both players 2 and 3 to not use the edge  $B \rightarrow C$ , but leaves player 1 with a cost of 9. In contrast, if player 1 can commit to a mixed strategy, she can randomize between using the path extended on the left side ( $S_1 \rightarrow A \rightarrow B \rightarrow C \rightarrow T_1$ )  $\varepsilon$  of the time, the path extended on the right side ( $S_1 \rightarrow B \rightarrow C \rightarrow D \rightarrow T_1$ )  $\varepsilon$  of the time, and the shortest path ( $S_1 \rightarrow B \rightarrow C \rightarrow T_1$ ) the rest of the time. This is enough to incentivize both of the followers to not use the edge between  $B$  and  $C$ , causing the expected cost for player 1 to be  $2\varepsilon(4 + \varepsilon) + (1 - 2\varepsilon)(1 + 2\varepsilon) = 1 + 4\varepsilon - 2\varepsilon^2$ . Letting  $\varepsilon \rightarrow 0$ , we obtain  $MvP^{\text{ISD}}(\text{QAR}_{3 \text{ player}}) \geq 9$ . This modification can in fact be applied to the  $\text{VoPC}^{\text{ISD}}(\text{QAR}_{n \text{ player}})$  construction for any  $n$ , giving us  $MvP^{\text{ISD}}(\text{QAR}_{n \text{ player}}) \geq n^2$ .  $\square$

***k-nomial Atomic Routing.*** We next extend this result to the case where the edge costs scale as a general monomial,  $x^k$ , with the number of players using an edge. Thus, each edge  $e$  has a coefficient  $c_e$ , and if  $n_e$  players use that edge, each of them incurs a cost of  $c_e n_e^k$ . We will refer to this class of games as *KAR* (*k-nomial Atomic Routing*).

**Theorem 34.** *For any solution concept  $SC \in \{\text{ISD}, \text{BestNash}, \text{WorstNash}, \text{BestCorrelated}, \text{WorstCorrelated}\}$ , we have  $\text{VoPC}^{SC}(\text{KAR}_{n \text{ player}}) = \text{VoMC}^{SC}(\text{KAR}_{n \text{ player}}) = MvP^{SC}(\text{KAR}_{n \text{ player}}) = n^k$  (where  $n$  is the number of players).*

*Proof.* The proofs for  $\text{VoMC}^{\text{ISD}}(\text{KAR}_{n \text{ player}}) \leq n^k$ ,  $\text{VoPC}^{SC}(\text{KAR}_{n \text{ player}}) \leq n^k$  and  $MvP^{SC}(\text{KAR}_{n \text{ player}}) \leq n^k$  follow the same logic as in Theorem 33, replacing  $n^2$  with  $n^k$ .

We now prove  $\text{VoPC}^{\text{ISD}}(\text{KAR}_{n \text{ player}}) \geq n^k$ . Again, the game structure greatly resembles the structure in the earlier routing proofs, with a few minor changes. Again we increase the costs of the edges between the upper and lower sections. We need them to be at least  $n^k - 1$ . Finally, we need to make some corresponding adjustments to the coefficients in the lower section to obtain similar strategic effects. Figure 8.15 gives the construction for the three-player (two-follower) case for general  $k$ .

Let us now show that this game shows that  $VoPC^{ISD}(KAR_{3 \text{ player}}) \geq 3^k$ . Again, after removing the obviously dominated strategies that visit a node more than once, player 1 has a dominant strategy of  $S_1 \rightarrow B \rightarrow C \rightarrow T_1$ . Once we fix this strategy for player 1, player 3 has a dominant strategy, namely  $S_3 \rightarrow F \rightarrow D \rightarrow C \rightarrow B \rightarrow T_3$ . Once we fix this, player 2 has a best response of  $S_2 \rightarrow E \rightarrow A \rightarrow B \rightarrow C \rightarrow T_2$ . This ISD solution gives player 1 a cost of  $3^k + 2\varepsilon$ .

When we consider pure strategy commitment, if player 1 commits to the path  $S_1 \rightarrow A \rightarrow B \rightarrow C \rightarrow D \rightarrow T_1$ , then player 2 has a dominant strategy of  $S_2 \rightarrow F \rightarrow D \rightarrow T_2$ . This leads to player 3 having a best response of  $S_3 \rightarrow E \rightarrow A \rightarrow T_3$ , and results in a cost of  $1 + 4\varepsilon$  for player 1. It follows that  $VoPC^{ISD}(KAR_{3 \text{ player}}) \geq \frac{3^k + 2\varepsilon}{1 + 4\varepsilon}$  for any  $\varepsilon$ , hence by letting  $\varepsilon \rightarrow 0$ , we get  $VoPC^{ISD}(KAR_{3 \text{ player}}) \geq 3^k$ .

The general construction for  $VoPC^{ISD}(KAR_{n \text{ player}})$  follows exactly the same pattern as in the previous two proofs on routing, with the same changes described above. The ISD solution in this game has a cost of  $n^k + 2\varepsilon$ , and the best pure-strategy commitment results in a cost of  $1 + 4\varepsilon$  for player 1. Thus  $VoPC^{ISD}(KAR_{n \text{ player}}) \geq \frac{n^k + 2\varepsilon}{1 + 4\varepsilon}$  for any  $\varepsilon$ , hence by letting  $\varepsilon \rightarrow 0$ , we get  $VoPC^{ISD}(KAR_{n \text{ player}}) \geq n^k$ .  $VoMC^{ISD}(KAR_{n \text{ player}}) \geq n^k$  follows from  $VoPC^{ISD}(KAR_{n \text{ player}}) \geq n^k$  and Lemma 1.

Let us now prove  $MvP^{ISD}(KAR_{n \text{ player}}) \geq n^2$ . The construction for  $MvP^{ISD}(KAR_{n \text{ player}})$  resembles the previous constructions. Again, we take the construction for  $VoPC^{SC}(KAR_{n \text{ player}})$  and make a few changes to it. Namely, we increase the edge coefficients in the upper section of the graph (except those on the ISD path of  $S_1 \rightarrow B \rightarrow C \rightarrow T_1$ ) from  $\varepsilon$  to  $\frac{n^k - 1}{4}$ . To compensate for this, we also need to make some changes to the coefficients in the bottom section. We illustrate this in Figure 8.16 for the 3-player case.

Again, this construction causes commitment to a pure strategy to hardly help player 1 at all. Player 1's best pure strategy commitment is still the path  $S_1 \rightarrow A \rightarrow B \rightarrow C \rightarrow DT_1$ , which will be enough to incentivize both players 2 and 3 to not use the edge  $B \rightarrow C$ , but

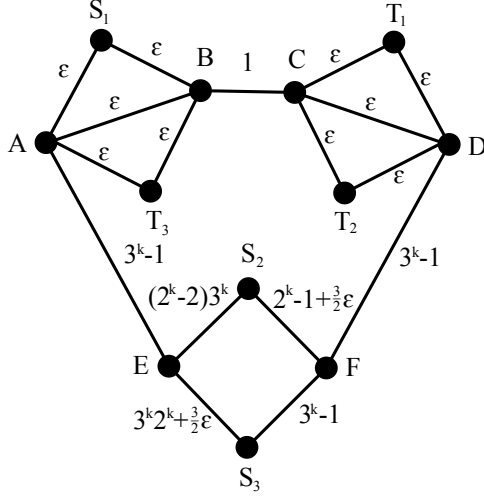


FIGURE 8.15: 3-player *KAR* game for *VoPC* and *VoMC*

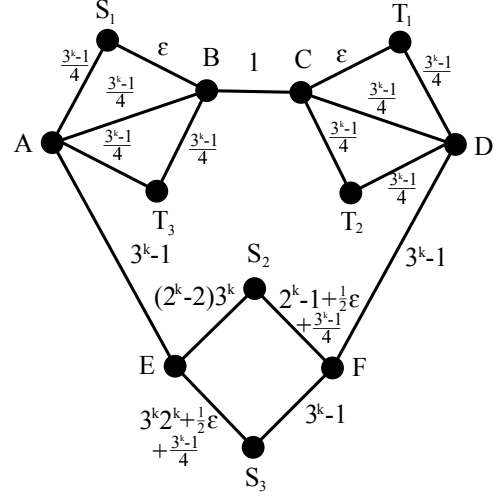


FIGURE 8.16: 3-player *KAR* game for *MvP*

leaves player 1 with a cost of  $3^k$ . In contrast, if player 1 can commit to a mixed strategy, she can randomize between using the path extended on the left side ( $S_1 \rightarrow A \rightarrow B \rightarrow C \rightarrow T_1$ )  $\varepsilon$  of the time, the path extended on the right side ( $S_1 \rightarrow B \rightarrow C \rightarrow D \rightarrow T_1$ )  $\varepsilon$  of the time, and the shortest path ( $S_1 \rightarrow B \rightarrow C \rightarrow T_1$ ) the rest of the time. This is enough to incentivize both of the followers to not use the edge between  $B$  and  $C$ , causing the expected cost for player 1 to be  $2\varepsilon(4 + \varepsilon) + (1 - 2\varepsilon)(1 + 2\varepsilon) = 1 + 4\varepsilon - 2\varepsilon^2$ . Letting  $\varepsilon \rightarrow 0$ , we obtain  $MvP^{ISD}(KAR_{3 \text{ player}}) \geq 3^k$ . This modification can in fact be applied to the  $VoPC^{ISD}(KAR_n \text{ player})$  construction for any  $n$ , giving us  $MvP^{ISD}(KAR_n \text{ player}) \geq n^k$ .  $\square$

### 8.7 Arbitrary cost atomic routing

Finally, let us consider the case where each edge has an arbitrary monotonically increasing cost based on the number of players using the edge. We will refer to this class of games as *AAR* (Arbitrary Atomic Routing). Here, we get arbitrarily large ratios even with two players.

**Theorem 35.** For any solution concept  $SC \in \{ISD, BestNash, WorstNash, BestCorrelated,$

WorstCorrelated}, we have  $VoPC^{SC}(AAR_{2\text{ player}}) = VoMC^{SC}(AAR_{2\text{ player}}) = MvP^{SC}(AAR_{2\text{ player}}) = \infty$ .

*Proof.* First, let us prove that  $VoPC^{ISD}(AAR_{2\text{ player}}) = \infty$ . Consider the game in Figure 8.17, where all edges but the edge between  $B \rightarrow T$  have linear costs (indicated with a coefficient as before), and  $B \rightarrow T$  has a cost of  $\epsilon$  if one person uses it, but a cost of 1 if both players use it.

Again, after removing the obviously dominated strategies that visit a node more than once, player 1 has a dominant strategy of  $S_1 \rightarrow B \rightarrow T$ . Given this path for player 1, the best-response strategy for player 2 is to take the path  $S_2 \rightarrow A \rightarrow B \rightarrow T$ . This ISD solution gives player 1 a cost of  $1 + \epsilon$ .

When we consider pure strategy commitment, if player 1 commits to the path  $S_1 \rightarrow A \rightarrow B \rightarrow T$ , then player 2's best response to is to take the alternate direct path  $S_2 \rightarrow T$ . This results in a cost of  $1 + 2\epsilon$  for player 1. It follows that  $VoPC^{ISD}(AAR_{2\text{ player}}) \geq \frac{1+\epsilon}{2\epsilon}$ , hence by letting  $\epsilon \rightarrow 0$ , we get  $VoPC^{ISD}(AAR_{2\text{ player}}) \geq \infty$ .  $VoMC^{ISD}(AAR_{2\text{ player}}) \geq \infty$  follows from  $VoPC^{ISD}(AAR_{2\text{ player}}) \geq \infty$  and Lemma 1.

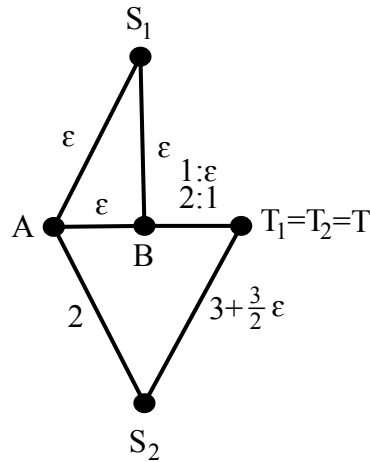


FIGURE 8.17: 2-player  $AAR_{2\text{ player}}$  game for  $VoPC$  and  $VoMC$

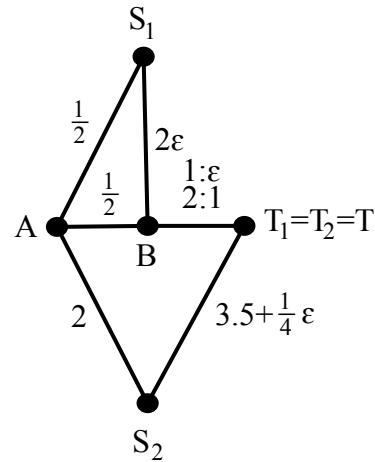


FIGURE 8.18: 2-player  $AAR_{2\text{ player}}$  game for  $MvP$

Let us now prove  $MvP^{ISD}(AAR_{2\text{ player}}) = \infty$ . We will use the game in Figure 8.18,

which is a slightly modified version of the game in Figure 8.17. Again, this construction causes commitment to a pure strategy to hardly help player 1 at all. player 1's best pure-strategy commitment is still the path  $S_1 \rightarrow A \rightarrow B \rightarrow T$ , which will be enough to incentivize player 2 to not use the edge  $B \rightarrow T$ , but leaves player 1 with a cost of  $1 + \varepsilon$ . In contrast, if player 1 can commit to a mixed strategy, she can randomize between  $S_1 \rightarrow A \rightarrow B \rightarrow T$   $\varepsilon$  of the time, and the shortest path ( $S_1 \rightarrow B \rightarrow T$ ) the rest of the time. This is enough to incentivize player 2 to not use the edge between  $B$  and  $T$ , causing the expected cost for player 1 to be  $\varepsilon(1 + \varepsilon) + (1 - \varepsilon)(2\varepsilon) = 3\varepsilon - \varepsilon^2$ . Letting  $\varepsilon \rightarrow 0$ , we obtain  $MvP^{\text{ISD}}(AAR_{2 \text{ player}}) = \infty$ . □

## Conclusion

This dissertation studied the computational problems involved in finding a Stackelberg equilibrium over a wide range of game representations. Our main focus was to show which assumptions in each of these representations are necessary for there to exist polynomial-time algorithms, by finding these algorithms and when possible, showing that when these assumptions are relaxed, these problems become NP-hard.

In Chapter 2 we considered computing approximately optimal Stackelberg strategies in Bayesian games. Our results here were mostly negative: we showed that the best possible approximation ratio that can be obtained in polynomial time for the standard Bayesian problem is  $O(t)$ , the number of types, unless  $NP = P$ . Optimizing for the worst type is completely inapproximable in polynomial time, in the sense that we cannot distinguish instances where we can guarantee utility 1 from instances where it is impossible to guarantee positive utility unless  $P=NP$ . We also studied a different representation of uncertainty about the followers payoffs that relies on ranges, and showed that optimizing for the worst case is NP-hard in the basic setting, and completely inapproximable in a generalized setting where the payoffs are linked. These negative results provide some justification for the use of worst-case exponential-time algorithms in this context, such as those that use mixed

integer programming.

In Chapter 3 we studied commitment in the more general setting of extensive-form games. We first described in detail a number of different modeling decisions that can be made when modeling a game in the extensive form. We then show how these modeling decisions effect the computational hardness of the problem. We first gave polynomial time algorithms for four cases and then showed that when we modify the assumptions made in these cases the problem usually becomes NP-hard.

In Chapter 4 we united two lines of research, the computation of optimal strategies to commit to (Stackelberg strategies), and the computation of correlated equilibria of stochastic games by studying the computation of Stackelberg strategies in stochastic games. We showed that the value of being able to commit and the value of being able to correlate are both unbounded. Additionally, we showed that removing the ability to correlate or the ability to use the full history of the game causes solving for the optimal commitment strategy to become NP-hard. Finally, we showed how to modify the QPACE algorithm [41] to compute approximately optimal Stackelberg strategies.

In Chapter 5 we focused on how the structure of possible schedules affects the computational hardness. Our approach is similar to that of Korzhyk et al. at a high level, but it turns out that for one of these problems, we need a generalization of the BvN theorem that was given by Budish et al. (2013) (which we gave a brief overview of in Section 5.1). We were able to use this generalization of the BvN theorem to characterize a new class of games where a mixed strategy corresponding to the marginal probabilities is guaranteed to exist, and gave us an algorithm for finding it. For another class of games, we showed how to compute optimal marginal probabilities in a specific way, and how to directly obtain a mixed strategy that corresponds to these marginal probabilities. We also showed that in a number of related settings there exists an instance where the optimal marginal solution does not correspond to any mixed strategy and in most of these cases the problem of finding an optimal Stackelberg strategy is in fact NP-hard.

In Chapter 6 we extended previous linear programming techniques to incorporate benefits and costs of arbitrary security configurations on individual assets. Second, we offered a principled model of failure cascades that allows us to capture both the direct and indirect value of assets, and showed how to extend this model to capture uncertainty about the structure of the interdependency network. Third, we allowed the defender to account for failures due to actual attacks, as well as those that are a result of exogenous failures. Our experimental results demonstrated the value of our approach as compared to alternatives, and showed that it is scalable to realistic security settings. Furthermore, we used our framework to analyze four models of interdependencies: two based on random graph generation models, a simple model of interdependence between critical infrastructure and key resource sectors, and a model of the Fedwire interbank payment network.

In Chapter 7 we presented a Stackelberg game model of security in which the defender chooses optimal mitigations that reduce the capability of the attacker to achieve his goals. We offered an integer programming formulation of this problem, and presented several variants of constraint generation, leveraging a state-of-the-art AI planning tool to dramatically increase scalability. Additionally, we presented a heuristic alternative, and showed that it uses far less memory than the optimal plan interdiction algorithm, yields nearly optimal solutions, and runs considerably faster than the optimal algorithm on some, though not all, test problems. We extended the classical planning framework to incorporate uncertainty about attacker's capabilities, costs, goals, as well as execution uncertainty, and showed that these extensions retain the basic structure of the deterministic plan interdiction problem and, therefore, its scalability. More generally, we provided an integer programming formulation for computing optimal interdiction strategies for MDPs, but generality here comes at a substantial cost to scalability.

Finally, in Chapter 8 we analyzed how much benefit a player can derive from commitment in various types of games in a quantitative sense. On one hand, the notion of the value of commitment studied in this chapter fits well among various other notions that aim

to quantify strategic effects in games, including the price of anarchy [31, 46], the price of stability [54], the value of mediation [4], etc. On the other hand, to our knowledge, this concept is unique among these concepts in the sense that it focuses on the benefit of a strategic tool—commitment—to a specific player, rather than the cost of strategic behavior to welfare in general. (It should be noted that the value of mediation also considers the benefit of a strategic tool—correlated strategies—to social welfare, though this is a somewhat different type of strategic tool, one that is used by the players collectively rather than by an individual player.)

On a higher level, I believe that there are a number of settings where Stackelberg techniques can provide reasonable, and perhaps equally importantly, computationally tractable solutions. While some of these settings have already been discovered, including the applications discussed earlier, work continues to improve both the quality of the solutions and the scale of problems that can be solved in these settings. While the correct balance between abstraction (for tractability) and faithfulness to the original problem (for the solutions found to be meaningful) is often specific to the application being considered, there are often trends and general techniques that suggest what assumptions may be necessary. Much of the work in the second part of this dissertation focused on graph or network domains because of the abstraction power of such domains. However, there remain many open questions, even in these domains. For example, to what other graph or network domains do these techniques extend? What are other powerful abstractions that would be useful for a wide range of applications? How well do our proposed models scale and what is the quality of the solutions in actual cybersecurity domains and how exactly do we evaluate this? For existing applications, in what ways can we improve the solution quality while maintaining or improving the tractability of the current techniques? There are also potential direct extensions of this work, for example, extending the work on plan interdiction (discussed in Chapter 7) to mixed strategies and improving the support for uncertainty in this domain.

# Bibliography

- [1] Adler, I. (2013), “The equivalence of linear programs and zero-sum games,” *International Journal of Game Theory*, 42, 165–177.
- [2] Albert, R., Jeong, H., and Barabasi, A.-L. (2000), “Error and attack tolerance of complex networks,” *Nature*, 406, 378–382.
- [3] Anshelevich, E., Dasgupta, A., Kleinberg, J., Tardos, É., Wexler, T., and Roughgarden, T. (2004), “The Price of Stability for Network Design with Fair Cost Allocation,” in *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 295–304.
- [4] Ashlagi, I., Monderer, D., and Tennenholtz, M. (2008), “On the Value of Correlation,” *Journal of Artificial Intelligence Research*, 33, 575–613.
- [5] Aumann, R. (1974), “Subjectivity and Correlation in Randomized Strategies,” *Journal of Mathematical Economics*, 1, 67–96.
- [6] Bertsimas, D. and Tsitsiklis, J. N. (1997), *Introduction to Linear Optimization*, Athena Scientific.
- [7] Birkhoff, G. (1946), “Tres observaciones sobre el algebra lineal,” *Univ. Nac. Tucumán Rev, Ser. A, no. 5*, pp. 147–151.
- [8] Budish, E., koo Che, Y., Kojima, F., and Milgrom, P. (2013), “Designing Random Allocation Mechanisms: Theory and Applications,” *American Economic Review*, 103, 585–623.
- [9] Chen, X. and Deng, X. (2006), “Settling the complexity of two-player Nash equilibrium,” in *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 261–272.
- [10] Chen, Y., Wah, B. W., and wei Hsu, C. (2006), “Temporal planning using subgoal partitioning and resolution in SGPlan,” *Journal of Artificial Intelligence Research*, 26, 323–369.
- [11] Christodoulou, G. and Koutsoupias, E. (2005), “On the price of anarchy and stability of correlated equilibria of linear congestion games,” in *Proceedings of the 13th Annual European Symposium*, pp. 59–70.

- [12] Conitzer, V. and Korzhyk, D. (2011), “Commitment to Correlated Strategies,” in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 632–637, San Francisco, CA, USA.
- [13] 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.
- [14] Conitzer, V. and Sandholm, T. (2008), “New Complexity Results about Nash Equilibria,” *Games and Economic Behavior*, 63, 621–641.
- [15] Dantzig, G. (1951), “A proof of the equivalence of the programming problem and the game problem,” in *Activity Analysis of Production and Allocation*, ed. T. Koopmans, pp. 330–335, John Wiley & Sons.
- [16] Daskalakis, C., Goldberg, P., and Papadimitriou, C. H. (2006), “The Complexity of Computing a Nash Equilibrium,” in *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, pp. 71–78.
- [17] Dodds, P. S. and Watts, D. J. (2005), “A generalized model of social and biological contagion,” *Journal of Theoretical Biology*, 232, 587–604.
- [18] Etessami, K. and Yannakakis, M. (2007), “On the Complexity of Nash Equilibria and Other Fixed Points (Extended Abstract),” in *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 113–123.
- [19] Filar, J. and Vrieze, K. (1997), *Competitive Markov Decision Processes*, Springer-Verlag.
- [20] Ghare, P., Montgomery, D., and Turner, W. (1971), “Optimal interdiction policy for a flow network,” *Naval Research Logistics Quarterly*, 18, 37–45.
- [21] Gilboa, I. and Zemel, E. (1989), “Nash and correlated equilibria: Some complexity considerations,” *Games and Economic Behavior*, 1, 80–93.
- [22] Jain, M., Pita, J., Tambe, M., Ordóñez, F., Paruchuri, P., and Kraus, S. (2008), “Bayesian Stackelberg games and their application for security at Los Angeles International Airport,” *SIGecom Exch.*, 7, 1–3.
- [23] Jain, M., Tambe, M., and Kiekintveld, C. (2011), “Quality-bounded Solutions for Finite Bayesian Stackelberg Games: Scaling up,” in *International Conference on Autonomous Agents and Multiagent Systems*.
- [24] Kempe, D., Kleinberg, J. M., and Éva Tardos (2003), “Maximizing the Spread of Influence in a Social Network,” in *Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 137–146.

- [25] 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.
- [26] Korilis, Y. A., Lazar, A. A., and Orda, A. (1997), “Achieving Network Optima using Stackelberg Routing Strategies,” *IEEE/ACM Transactions on Networking*, 5, 161–173.
- [27] 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.
- [28] Korzhyk, D., Conitzer, V., and Parr, R. (2011a), “Security Games with Multiple Attacker Resources,” in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 273–279, Barcelona, Catalonia, Spain.
- [29] Korzhyk, D., Conitzer, V., and Parr, R. (2011b), “Solving Stackelberg Games with Uncertain Observability,” in *Proceedings of the Tenth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Taipei, Taiwan.
- [30] Korzhyk, D., Yin, Z., Kiekintveld, C., Conitzer, V., and Tambe, M. (2011c), “Stackelberg vs. Nash in Security Games: An Extended Investigation of Interchangeability, Equivalence, and Uniqueness,” *Journal of Artificial Intelligence Research*, 41, 297–327.
- [31] Koutsoupias, E. and Papadimitriou, C. H. (1999), “Worst-case equilibria,” in *Symposium on Theoretical Aspects in Computer Science*, pp. 404–413.
- [32] Kuhn, H. W. (1953), “Extensive Games and the Problem of Information,” in *Contributions to the Theory of Games*, eds. H. W. Kuhn and A. W. Tucker, vol. 2 of *Annals of Mathematics Studies*, 28, pp. 193–216, Princeton University Press.
- [33] Kunreuther, H. and Heal, G. (2003), “Interdependent Security,” *Journal of Risk and Uncertainty*, 26, 231–249.
- [34] 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.
- [35] Letchford, J. and Conitzer, V. (2013), “Solving security games on graphs via marginal probabilities,” in *To appear in Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI-13)*.
- [36] Letchford, J. and Vorobeychik, Y. (2012), “Computing Optimal Security Strategies for Interdependent Assets,” in *Conference on Uncertainty in Artificial Intelligence (UAI)*, Catalina Island, CA, US.

- [37] Letchford, J. and Vorobeychik, Y. (2013), “Optimal Interdiction of Attack Plans,” in *To appear in Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.
- [38] Letchford, J., Conitzer, V., and Munagala, K. (2009), “Learning and Approximating the Optimal Strategy to Commit To,” in *Proceedings of the 2nd International Symposium on Algorithmic Game Theory, SAGT '09*, pp. 250–262, Berlin, Heidelberg, Springer-Verlag.
- [39] 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 AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1380–1386, Toronto, ON, Canada.
- [40] Letchford, J., Korzhyk, D., and Conitzer, V. (2013), “On the Value of Commitment,” Working paper.
- [41] MacDermed, L., Narayan, K. S., Isbell, C. L., and Weiss, L. (2011), “Quick Polytope Approximation of All Correlated Equilibria in Stochastic Games,” in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, San Francisco, CA, USA.
- [42] McMasters, A. and Mustin, T. (1970), “Optimal interdiction of a supply network,” *Naval Research Logistics Quarterly*, 17, 261–268.
- [43] Mounzer, J., Alpcan, T., and Bambos, N. (2010), “Integrated Security Risk Management for IT-Intensive Organizations,” in *Sixth International Conference on Information Assurance and Security*, pp. 329–334.
- [44] Murray, C. and Gordon, G. (2007), “Finding correlated equilibria in general sum stochastic games,” Tech. Rep. CMU-ML-07-113, Carnegie Mellon University.
- [45] Newman, M. (2010), *Networks: An Introduction*, Oxford University Press.
- [46] Papadimitriou, C. H. (2001), “Algorithms, games and the Internet,” in *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, pp. 749–753.
- [47] 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.
- [48] 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.
- [49] Pita, J., Jain, M., Ordóñez, F., Tambe, M., and Kraus, S. (2010), “Robust Solutions to Stackelberg Games: Addressing Bounded Rationality and Limited Observations in Human Cognition,” *Artificial Intelligence*, 174, 1142–1171.

- [50] Pita, J., Tambe, M., Kiekintveld, C., Cullen, S., and Steigerwald, E. (2011), “GUARDS - Game Theoretic Security Allocation on a National Scale,” in *Proceedings of the Tenth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Taipei, Taiwan.
- [51] Roughgarden, T. (2004), “Stackelberg Scheduling Strategies,” *SIAM Journal on Computing*, 33, 332–350.
- [52] Roughgarden, T. and Tardos, É. (2002), “How Bad is Selfish Routing?” *Journal of the ACM*, 49, 236–259.
- [53] Salmeron, J., Wood, K., and Baldrick, R. (2009), “Worst-Case Interdiction Analysis of Large-Scale Electric Power Grids,” *IEEE Transactions on Power Systems*, 24, 96–104.
- [54] Schulz, A. S. and Moses, N. S. (2003), “On the performance of user equilibria in traffic networks,” in *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 86–87, Society for Industrial and Applied Mathematics.
- [55] Shieh, E., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., Maule, B., and Meyer, G. (2012), “PROTECT: A Deployed Game Theoretic System to Protect the Ports of the United States,” in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 847–854.
- [56] Soramaki, K., Bech, M. L., Arnold, J., Glass, R. J., and Beyeler, W. (2007), “The topology of interbank payment flows,” *Physica A*, 379, 317–333.
- [57] Tsai, J., Rathi, S., Kiekintveld, C., Ordonez, F., and Tambe, M. (2009), “IRIS - A Tool for Strategic Security Allocation in Transportation Networks,” in *The Eighth International Conference on Autonomous Agents and Multiagent Systems - Industry Track*, pp. 37–44.
- [58] Tsai, J., Nguyen, T. H., and Tambe, M. (2012), “Security games for controlling contagion,” in *Twenty-Sixth National Conference in Artificial Intelligence*.
- [59] University of Oregon (2011), “University of Oregon Route Views Project, Online Data and Reports,” <http://www.routeviews.org>.
- [60] van den Briel, M., Sanchez, R., Do, M. B., and Kambhampati, S. (2004), “Effective approaches for partial satisfaction (over-subscription) planning,” in *Nineteenth National Conference on Artificial Intelligence*, pp. 562–569.
- [61] von Neumann, J. (1928), “Zur Theorie der Gesellschaftsspiele,” *Mathematische Annalen*, 100, 295–320.
- [62] von Stackelberg, H. (1934), *Marktform und Gleichgewicht*, pp. 58–70, Springer, Vienna.

- [63] von Stengel, B. and Forges, F. (2008), “Extensive form correlated equilibrium: Definition and computational complexity,” *Mathematics of Operations Research*, 33, 1002–1022.
- [64] von Stengel, B. and Zamir, S. (2010), “Leadership Games with Convex Strategy Sets,” *Games and Economic Behavior*, 69, 446–457.
- [65] Vossen, T., Ball, M., and Smith, R. H. (1999), “On the Use of Integer Programming Models in AI Planning,” in *Sixteenth International Joint Conference on Artificial Intelligence*, pp. 304–309.
- [66] Yang, R., Ordonez, F., and Tambe, M. (2012), “Computing Optimal Strategy against Quantal Response in Security Games,” in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.
- [67] Yin, Z., Jiang, A. X., Johnson, M., Kiekintveld, C., Leyton-Brown, K., Sandholm, T., Tambe, M., and Sullivan, J. (2012), “TRUSTS: Scheduling Randomized Patrols for Fare Inspection in Transit Systems,” in *Proceedings of Innovative Applications of Artificial Intelligence (IAAI)*.
- [68] Zhuang, J. and Bier, V. (2007), “Balancing Terrorism and Natural Disasters—Defensive Strategy with Endogenous Attacker Effort,” *Operations Research*, 55, 976–991.
- [69] Zuckerman, D. (2007), “Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number,” *Theory of Computing*, 3, 103–128.

# Biography

Joshua Letchford was born in Bettendorf, Iowa, United States on May 16, 1984. He received a B.S in Computer Engineering from the University of Southern California in 2006. From 2007 to 2013, he studied at Duke University for a Ph.D. in Computer Science.

Joshua Letchford's research focuses on computational aspects of Stackelberg games. His research interests include computational economics, algorithmic game theory, multi-agent systems and social networks.