Scaling Empirical Game-Theoretic Analysis

by

Ben-Alexander Cassell

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2014

Doctoral Committee:

Professor Michael P. Wellman, Chair
Professor John E. Laird
Professor Jeffrey K. MacKie-Mason
Professor Demosthenis Teneketzis

2014

To Dad, Mom, Mary Rose, and Josh.

Acknowledgments

I would first like to extend my deepest thanks to my advisor, Michael Wellman. Michael always seemed to know exactly when I needed encouragement, and when I needed to be pushed, to bring out the best in me. He gave me the flexibility to pursue the topics that I found most interesting while providing invaluable guidance. I would also like to thank my dissertation committee, John Laird, Jeffrey MacKie-Mason, and Demosthenis Teneketzis, without whom this work would not be possible. I consider each member of my committee to be extraordinary scientists, teachers, and friends. I also owe a large amount of gratitude to Carey Bagdassarian at the College of William and Mary, my first research mentor, for encouraging me to aim high with my ambitions.

I am deeply thankful to the administrative staff of the Computer Science and Engineering department. In particular, I must thank Dawn Freysinger, Rita Rendell, and Cindy Watts, who always greeted me with a smile and a determination to resolve my administrative challenges. I would also like to thank DCO, and especially Laura Fink, for helping me manage the servers that have been instrumental to my research.

The rigors of the PhD program would have been more daunting were it not for a group of amazing friends and colleagues. I would like to especially thank the friends who put up with living with me over the years, and who are the source of some of my fondest graduate school memories: Daniel Fabbri, Jamie Kidwell, Erin Payne, Quang Duong, David Meisner, Steven Pelley, and Michael Chow. I am deeply indebted to my lab-mate Patrick Jordan, who set a sterling example of hard work and dedication, and demonstrates an unwavering belief in my ability, even when I question it myself. I would also like to thank Andrea Jordan, who has always treated me like family. I would like to thank my friends/collaborators, Timur Alperovich and Bryce Wiedenbeck, for always stimulating my brain with interesting discussion. I would be remiss if I did not acknowledge the past and present members of the Strategic Reasoning Group for their camaraderie and support. I must also thank my longtime friends from my hometown, St. Louis: Tony Flesor, Dan Pritt, Brandon Walsh, and Sarah Brandt. I would not have gotten this far without their love.

I must thank my parents, Stu Cassell, Mary Rose Cassell, and Carol Rutherford, for always supporting me in my educational endeavors. The pride they have in me is contagious, bolstering me when I am low and sharing in the joy when I succeed. Finally, I wish to thank my brother and best friend, Josh.

TABLE OF CONTENTS

Dedication . . . . . **ii**

Acknowledgments . . . . . **iii**

List of Figures . . . . . **vii**

List of Tables . . . . . **viii**

Chapter

1 Introduction **1**

1.1 Empirical Game-Theoretic Framework 2
1.1.1 Strategic Games 2
1.1.2 Empirical Game Models 3
1.1.3 Solution Concepts 4
1.1.4 Scaling EGTA 4
1.2 Overview of Contributions 7
1.2.1 Application: Wireless Access Point Selection 8
1.2.2 EGTAOnline: Software Infrastructure for Experiment Management 8
1.2.3 Efficient Analysis of Large Game Data Sets 8
1.2.4 Bootstrap Methods for Sequential Estimation of Nash Equilibria . 9
1.2.5 Application: Equity Premium Estimation in Asset Pricing 9
1.3 Guide to Reading this Thesis 10

2 Application: Wireless Access Point Selection **11**

2.1 Related Work 12
2.2 Game Description 13
2.2.1 Multiple AP Selection 13
2.2.2 Information Models 14
2.3 Strategies 15
2.3.1 Association Policies 15
2.3.2 Probing Policies 17
2.4 Experiments 18
2.5 Results 19
2.5.1 Bulletin Board Model 19
2.5.2 Probing Model 20
2.5.3 Social Welfare 22
2.6 Summary: AP Selection Game 23

2.7 Scaling Lessons . . . 23

3 EGTAOnline: Software Infrastructure for Experiment Management . . . 25

3.1 Related Work . . . 26
3.2 Role Symmetry . . . 27
3.3 Data Compatibility . . . 28
3.4 Architecture . . . 29
3.4.1 Simulators . . . 29
3.4.2 Observations . . . 30
3.4.3 Schedulers . . . 31
3.4.4 Simulations . . . 33
3.4.5 Profiles . . . 34
3.4.6 Games . . . 34
3.5 Data Reuse . . . 34
3.6 Automated Refinement of Game Models . . . 37
3.6.1 Exploration of Profile Space . . . 38
3.6.2 Sequential Estimation of Empirical Games . . . 39
3.7 In Production . . . 41

4 Efficient Analysis of Large Game Data Sets . . . 42

4.1 Background: Database Management . . . 43
4.2 Representing Games in a Database . . . 44
4.3 Game-Theoretic Primitives . . . 46
4.4 Identifying PSNE . . . 48
4.5 PSNE-Finding Performance . . . 51
4.5.1 Comparison of SQL Algorithms . . . 51
4.5.2 Comparison to Gambit . . . 52
4.6 Incremental Analysis . . . 56
4.6.1 Incremental Regret Maintenance . . . 57
4.6.2 Identifying Maximal Complete Subgames . . . 57
4.7 Discussion . . . 60

5 Bootstrap Methods for Sequential Estimation of Nash Equilibria . . . 62

5.1 The Bootstrap . . . 64
5.2 Using the Bootstrap in Sample Control . . . 66
5.3 Experimental Data Sets . . . 68
5.4 Sequential Classification of Profiles as Nash Equilibria . . . 69
5.5 Sequential Search for δ-Equilibria . . . 72
5.6 Discussion . . . 76

6 Application: Equity Premium Estimation in Asset Pricing . . . 79

6.1 Background: Agent Modeling . . . 80
6.2 Ambiguity Aversion and the Equity Premium Puzzle . . . 82
6.3 Empirical Game Model of Asset Pricing . . . 83
6.3.1 Market and Asset Models . . . 83
6.3.2 Agent Strategy Composition . . . 84

6.3.3 Estimating the Empirical Game . . . 85
6.4 Simulation of Asset Pricing under Ambiguous Information . . . 87
6.4.1 Market Conditions . . . 87
6.4.2 News . . . 88
6.4.3 Traders . . . 88
6.5 Experiments . . . 90
6.5.1 Equilibrium Analysis . . . 91
6.5.2 Equity Premium Estimation . . . 93
6.6 Discussion . . . 96

7 Conclusion . . . 100

7.1 Contributions . . . 100
7.1.1 Software Systems and Methods . . . 100
7.1.2 Applications . . . 103
7.2 Future Work . . . 104
7.3 Final Remarks . . . 105

Bibliography . . . 106

LIST OF FIGURES

1.1 Iterative, search-based approach to empirical game-theoretic analysis. 6

2.1 Lowest expected delay among found Nash equilibria for different rates of work clearing. 22

3.1 Supporting the EGTA process with EGTAOnline. 30
3.2 Fraction of unreduced-game profile space for an n-player reduction that is covered by smaller reductions, shown for varying size of strategy set, M. . . . 37
3.3 Implementing a sequential estimation procedure with EGTAOnline. 40

4.1 Schema for empirical games. 45
4.2 Data dependence graph for calculating regret of profiles in an example game. . 49
4.3 Finding best responses in the symmetric aggregations table. 51
4.4 Performance of two SQL methods for identifying all PSNE. 52
4.5 Comparing performance of PSNE finding methods under two cache scenarios. 55

6.1 Workflow diagram for using EGTA to estimate a non-payoff variables V. . . . 84
6.2 Abstract view of market simulation. 87

7.1 Idealized automated empirical game-theoretic analysis. 101

LIST OF TABLES

2.1 Equilibrium analysis of the bulletin board model. 19
2.2 Equilibrium analysis of the probing model. 21

5.1 Sequential classification performance. 71
5.2 Stopping rule performance. 75

6.1 Base market configuration for experiments. 91
6.2 Variant market configurations tested. 92
6.3 Symmetric mixed equilibria found in each of the market configurations. 92
6.4 Equity premium statistics for each market configuration. 95
6.5 Equity premium measurements for each market configuration when a single ambiguity-averse pricing strategy is played. 97

CHAPTER 1

Introduction

From the interplay of traders in financial markets to the interplay of armies at war, large-scale strategic interactions have a significant impact on our daily lives. To analyze strategic interaction, such scenarios are often cast as *games*, where the participants are players seeking to optimize their payoffs through the selection of a strategy in anticipation of the strategic decisions of other players. For complex games, such as those that might capture incentives of a financial market or a war, analytical models of the scenario may ignore significant features of the system in the name of tractability. When our modeling needs are too complex to address analytically, *empirical game models* (Wellman, 2006), where observations or simulated play are used to estimate the utility functions of agents, can be employed to facilitate game-theoretic analysis. The set of methods for building and analyzing empirical game models is known collectively as *empirical game-theoretic analysis* (EGTA).

Employing EGTA is not without its own limitations, as increasing the complexity of the model may require observing a greater number of outcomes, taking larger samples of these outcomes, and increasing the computational cost of analysis. As such, though EGTA affords more complex models than analytical approaches, there remains a frontier of computational feasibility. This frontier constrains the number of players, variety of strategies, and environment sophistication of empirical game models, reducing the fidelity of the model, and potentially hampering its ability to predict or describe real-world outcomes.

This thesis addresses the challenge of extending the boundaries of the EGTA framework to model and analyze large games. To this end, I present algorithms and software architectures that facilitate efficient construction, management, and analysis of large game models. In contrast to previous research into scaling EGTA, I place particular emphasis on managing the copious data generated by this methodology. Prior to providing an overview of the contents of this thesis, I present necessary definitions and background for the EGTA methodology.

1.1 Empirical Game-Theoretic Framework

1.1.1 Strategic Games

A game describes a strategic decision scenario and is commonly represented in *normal form*, $\Gamma = \langle I, \{S_i\}, \{u_i\}\rangle$, where I is the set of players, S_i is the non-empty set of strategies available to i, and u_i is a utility function that maps from $s \in \times_{i\in I} S_i$, a joint strategy or *pure-strategy profile*, to the utility that i receives when s is played. Where convenient, $N = |I|$ is used to express the number of players in a game and $M = |S|$ to indicate the size of the strategy set. I focus exclusively on non-cooperative games: games in which players are rational, self-interested agents that cannot make binding agreements to cooperate. A normal-form game can be equivalently expressed as an N-dimensional *payoff matrix*, where each dimension corresponds to the strategic choice of one player and each entry in the matrix corresponds to the vector of payoffs, $(u_1(s), ..., u_N(s))$, that players receive when playing the profile $s = (s_1, \ldots, s_N)$ that indexes that entry. A *subgame* of the game Γ, $\Gamma_X = \langle I, \{S'_i \subseteq S_i\}, \{u_i\}\rangle$, where $X = \times_{i\in I} S'_i$, refers to a game obtained by restricting the strategy sets of Γ, while retaining the players and projecting utility functions to the smaller profile space.

In game-theoretic analysis it is often useful to reason about the set of profiles that can be reached from profile s by a single player changing its strategy: the *unilateral deviation set*. Let s_i denote the strategy played by i in profile s and s_{-i} denote the joint strategy of all players other than i. The deviation set for player i from the profile s is given by $\mathcal{D}_i(s) = \{(\hat{s}_i, s_{-i}) : \hat{s}_i \in S_i \setminus \{s_i\}\}$. The deviation set of the profile s is given by $\mathcal{D}(s) = \bigcup_{i\in I} \mathcal{D}_i(s)$, and the deviation set of a set of profiles X is $\mathcal{D}(X) = \bigcup_{s\in X} \mathcal{D}(s)$.

Player i may also play a *mixed strategy*, σ_i, a probability mixture over its set of strategies, with the space of such mixtures denoted Δ_i. The set of strategies played with non-zero probability in σ_i is known as the *support* of σ_i, which I denote by $\mathcal{S}(\sigma_i)$. When one or more players play a mixed strategy, the joint strategy is referred to as a *mixed-strategy profile* and denoted σ. Just as a mixed strategy specifies the probability that a player will play one of its pure strategies, a mixed-strategy profile specifies the probability that each pure-strategy profile will be played. The support of σ, $\mathcal{S}(\sigma) = \times_{i\in I}\mathcal{S}(\sigma_i)$, is the space of pure-strategy profiles that are realized with non-zero probability when σ is played. The (von Neumann-Morgenstern) utility to i for playing its assignment in σ is given by the expected utility over the pure-strategy profiles that can be realized under the joint-strategy mixture,

$$u_i(\sigma) = \sum_{s\in\mathcal{S}(\sigma)} \Pr(s \mid \sigma) u_i(s),$$

where $\Pr(s \mid \sigma)$ is the probability that s is realized when σ is played:

$$\Pr(s \mid \sigma) = \prod_{s_i \in s} \Pr(s_i \mid \sigma)$$

As with pure-strategy profiles, we can define deviation sets for mixed-strategy profiles. It is sufficient for my purposes to restrict attention to deviations to pure strategies. Let σ_i denote the mixed-strategy played by i in the mixed-strategy profile σ, and σ_{-i} denote the joint strategy of players other than i. Then, $\mathcal{D}_i(\sigma) = \{(\hat{s}_i, \sigma_{-i}) : \hat{s}_i \in S_i \setminus \mathcal{S}(\sigma_i)\}$ and $\mathcal{D}(\sigma) = \bigcup_{i \in I} \mathcal{D}_i(\sigma)$.

1.1.2 Empirical Game Models

Empirical game models are built upon an underlying simulator or other model for generating observations. Simulators allow us to sample the outcome of agent play, providing a noisy estimate of each agent's utility when playing the observed joint strategy. Though the full space of possible strategies may be very large or infinite, we usually confine our attention to a small set of heuristic strategies. This set of strategies induces a space of profiles that is at best exponential in the smaller of the number of players and the number of strategies. As such, when an empirical game model features a large number of players and strategies, sampling the full space of profiles may be infeasible.

I focus on building game models by gathering observation data from simulators. Each observation corresponds to a single simulation, and records the vector of payoffs achieved by agents in that run of the simulator. An observation may also record other statistics of interest such as realizations of random variables or features of the simulation that arise through agent interaction, such as the winning bid in a simulated auction. I represent a set of observations by $\Theta = \{\theta\}$, where each observation θ specifies the profile played, the payoff vector observed, and any statistics recorded.

Models of the *true* or *underlying game*, terms which I use interchangeably to refer to the game being simulated, may be constructed from a set of compatible observations. I discuss in detail what it means for observations to be compatible in Chapter 3. Jordan and Wellman (2009) cover model selection extensively, providing a framework for evaluating goodness-of-fit of a game model to observation data. I primarily focus on a simple transformation of observation data into a payoff matrix: for every profile s for which we have at least one observation, $u_i(s) = \bar{u}_i(s)$, the average payoff that player i received in our observations of s. This payoff matrix is not strictly equivalent to a normal-form game, since utility functions will be undefined for unobserved profiles.

1.1.3 Solution Concepts

We construct models of agent utilities from observation data in order to predict agent behavior in the system being simulated. Such predictions are generated by a *solution concept*, a rule for predicting game play. The solution concept that is predominantly adopted in game-theoretic analysis is the *Nash equilibrium*. A Nash equilibrium is a (potentially mixed-strategy) profile such that no player can improve their payoff through unilaterally switching to a different strategy.

We often wish to identify the set of strategies that allow a player to *best respond*—achieve its highest possible payoff given the strategic choices of the other players. This set is identified by the *best-response correspondence*. Though in principle the strategies in a player's best-response correspondence can be mixed strategies, I limit attention to the pure-strategy best-response correspondence. For any profile σ and player i, the pure-strategy best-response correspondence is given by:

$$\mathcal{B}_i(\sigma_{-i}) = \arg\max_{\hat{s}_i \in S_i} u_i(\hat{s}_i, \sigma_{-i}).$$

We can also define the best-response correspondence of a profile as $\mathcal{B}(\sigma) = \times_{i \in I} \mathcal{B}_i(\sigma_{-i})$. A Nash equilibrium is a profile σ such that $\forall_i \mathcal{S}(\sigma_i) \subseteq \mathcal{B}_i(\sigma_{-i})$.

It is often helpful to identify profiles that are good approximations to Nash equilibrium in terms of player payoffs. To describe how well a profile σ approximates a Nash equilibrium, I use a measure called *regret*, notated $\epsilon(\sigma)$. Regret is the maximum payoff benefit that any single player can achieve through changing their strategy, while holding other players' strategies constant. Regret is given by:

$$\epsilon(\sigma) = \max_{i \in I} \max_{s_i \in S_i} u_i(s_i, \sigma_{-i}) - u_i(\sigma).$$

A profile σ is a Nash equilibrium if and only if it has regret equal to zero. More generally, we say that a profile σ is a δ-approximation of Nash equilibrium, or simply a δ-Nash equilibrium, if $\epsilon(\sigma) \leq \delta$.

1.1.4 Scaling EGTA

While EGTA provides a framework capable of analyzing the incentive structure of almost any strategic scenario that we can simulate, we are limited by the time spent in simulation and analysis. If we wish to understand a market with 100 participants and more than a few strategies for each participant, we cannot hope to do so by enumerating all the Nash

equilibria of the 100-player game. Assuming each player has 3 strategies, and the game has no useful underlying structure, estimating payoffs for all pure-strategy profiles of this game would require sampling the outcome of play from $3^{100} \approx 5 \times 10^{47}$ profiles. Furthermore, for each profile that we sample, we may need to take a large number of observations to ensure that our payoff estimates are accurate. Not only would this take an insurmountable simulation effort, finding all the equilibria of this game would also take considerable time, as Daskalakis et al. (2009) demonstrated that the problem of finding a mixed-strategy Nash equilibrium is PPAD-complete, and remarked that no known algorithm for finding equilibria has been proven to be polynomial.

To identify all (δ-)Nash equilibria of a game model, we must have payoff estimates for every pure-strategy profile. Any pure-strategy profile for which we do not have payoff estimates has the potential to be an equilibrium, and some unobserved profiles have the potential to refute candidate equilibria. It is often possible, however, to confirm a single equilibrium without sampling the full profile space. Since calculating $\epsilon(\sigma)$ requires evaluating player utilities for σ and single-player deviations from σ, we can confirm or reject σ as an equilibrium after sufficiently sampling all s in $\mathcal{S}(\sigma) \cup (\bigcup_{\hat{\sigma} \in \mathcal{D}(\sigma)} \mathcal{S}(\hat{\sigma}))$. To see how this set differs from the full profile space, consider the profile from the previously described 100-player game in which all players play their first strategy with probability 1. In this special case, σ is equivalent to a pure-strategy profile, so $\mathcal{S}(\sigma) = \{\sigma\}$ and $\mathcal{S}(\hat{\sigma}) = \{\hat{\sigma}\}$ for $\hat{\sigma} \in \mathcal{D}(\sigma)$. Since $|\mathcal{D}(\sigma)| = 200$, as each of the 100 players can deviate to one of their two other strategies, $|\mathcal{S}(\sigma) \cup (\bigcup_{\hat{\sigma} \in \mathcal{D}(\sigma)} \mathcal{S}(\hat{\sigma}))| = 201$, considerably fewer profiles than the 5×10^{47} profiles in the full space.

As many games of interest are too large to fully sample, and some games may take considerable time to gather a single observation,[1] much work conducted under the EGTA methodology focuses on confirming one or more δ-Nash equilibria, rather than identifying all Nash equilibria, concluding sampling after achieving this goal. This desire to be economical in sampling led Jordan et al. (2008) to formulate the problem of identifying approximate Nash equilibria as a search through profile space, yielding an iterative approach to sampling and game model construction, presented in Figure 1.1. In each iteration of this procedure a pure-strategy profile is selected for sampling, leading to an updated observation matrix. With this new observation matrix we can conduct game-theoretic or statistical analysis to determine whether further refinement of our game model is required. This refinement can take the form of sampling unobserved profiles, increasing the size of profile space by adding strategies, or through further sampling of previously sampled profiles to

[1] For example, the game simulator studied by Jordan et al. (2007) requires over 7 CPU hours to gather a single observation.

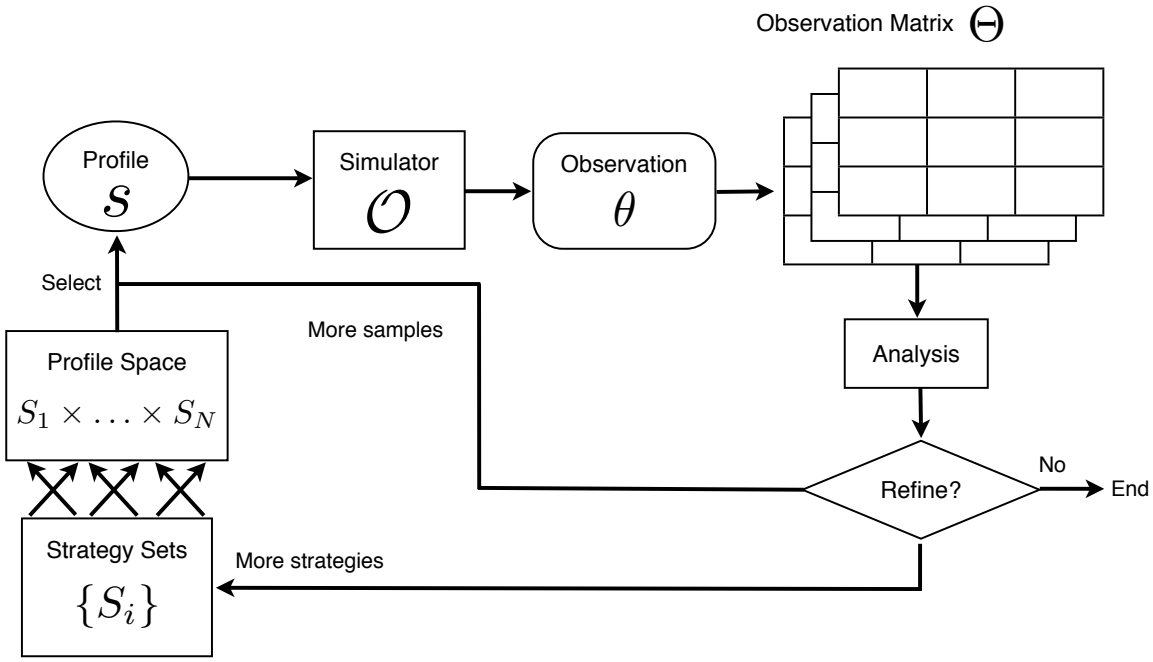


Figure 1.1: Iterative, search-based approach to empirical game-theoretic analysis.

reduce uncertainty in payoff estimates.

Often the space of profiles to sample can be reduced by leveraging symmetries in payoff structure. *Symmetric games* are games in which all players share a single strategy set and utility function that depends upon only the player's strategy and the counts of strategies played by others. For symmetric games, we can express the normal form representation as $\langle N, S, u(\cdot)\rangle$. In such games, the space of profiles that are unique with respect to player symmetry is much smaller than that of the non-symmetric representation (Cheng et al., 2004). Assuming the 100-player market game introduced above could be cast as a symmetric game, the number of profiles in the game shrinks to a much more manageable size: $\binom{100+3-1}{100} = 5151$. Reeves et al. (2005) found that game solvers incorporating symmetry could solve symmetric games much faster than solvers oblivious of symmetry. I develop a more general symmetric representation that I term *role symmetry*, presented in detail in Chapter 3

Another way to reduce the space of profiles to be sampled is through reasoning about large games with abstractions. Wellman et al. (2005) proposed the *hierarchical reduction*, a game abstraction for analyzing symmetric games in which groups of multiple agents are controlled by a single player. Under the hierarchical reduction we may choose to model a simulation with 100 agents as a 10-player game in which each player chooses the strategy to played by 10 agents. When payoffs vary smoothly as agents change strategies, solutions

to a hierarchically-reduced game can be good approximations to the solutions of the unreduced game. However, since a single player chooses the strategy played by several agents, a deviation by a player in the reduced game corresponds to multiple players deviating to the same strategy in the unreduced game. Accordingly, single-player deviation incentives, which form the basis of Nash equilibrium, may not be accurately represented in the reduced game. To address this concern, Wiedenbeck and Wellman (2012) proposed the *deviation-preserving reduction*. In the deviation-preserving reduction, each player views themselves as an individual, but view their opponents as an aggregate represented by some small number of players. Wiedenbeck and Wellman find this reduction superior, in terms of regret of solutions to the reduced game when played in the full game, to hierarchical reduction and the similar twins reduction (Ficici et al., 2008) for several classes of games. Both hierarchical reduction and deviation-preserving reduction require exponentially fewer profiles than the unreduced game, and thus reductions are commonly used for analyzing large empirical games.

1.2 Overview of Contributions

Although previous work has focused on reducing the space of profiles that must be sampled, the challenge of scale continues to limit which scenarios can be effectively explored. Prior to my development of software infrastructure specifically to address large games (see Chapter 3), the largest games routinely studied with EGTA were the various scenarios of the Trading Agent Competition (TAC), an academic trading agent design challenge, which range in size from six to eight players. Even when examining a game with only six players, analysis conducted on the TAC Supply Chain Management game required exploiting symmetry, utilizing hierarchical reduction, and considering only small sets of heuristic strategies to keep the simulation requirements manageable (Jordan et al., 2007). Improvements in our ability to efficiently explore large games through simulation not only let us analyze strategic scenarios with a large number of participants, but also let us consider a larger space of strategies, and can reduce the time spent waiting on data acquisition. To improve the ability of EGTA to scale to large games, I present algorithms and software infrastructure to manage the efficient acquisition of profile observations from a simulator such that δ-Nash equilibria of the derived game model can be identified and confirmed, to some specified level of statistical confidence. The remainder of this chapter introduces the content of my thesis, including three avenues for scaling EGTA and two applications.

1.2.1 Application: Wireless Access Point Selection

I begin with an application that predates the development of my scaling methods to introduce the reader to applying EGTA to real-world challenges. This application considers the problem of selecting a public wireless access point when multiple access points are available. Agents interact by virtue of their traffic congesting the available access points, leading to latency. My simulation model extends the traditional load-balancing game approach by having players engage in a dynamic game where work persists over time, as well as by modeling two emerging wireless technologies: the ability to use multiple access points simultaneously, and the use of probes to determine the level of congestion at access points.

1.2.2 EGTAOnline: Software Infrastructure for Experiment Management

As the size of the game grows, so does the volume of simulation required. At large scales, coordinating the required simulation and organizing the resulting data is too unwieldy to be carried out manually. To alleviate the cost to researchers of managing game simulation, I developed EGTAOnline, software infrastructure that automates significant portions of the process of constructing games from simulation. EGTAOnline can also dramatically improve throughput by distributing simulation to be run in parallel on a compute cluster. Users of EGTAOnline are spared from learning the orthogonal skill of writing scheduling scripts for the cluster, and instead can express complex scheduling requirements through a simple web interface. EGTAOnline also makes it easy for users to analyze subgames of interest, rather than downloading all of the available data, reducing the time spent in analysis. Further reducing simulation time and storage space requirements, through EGTAOnline I promote and support a compact representation of games that generalizes symmetry—role symmetry.

1.2.3 Efficient Analysis of Large Game Data Sets

With EGTAOnline, I have pushed the EGTA methodology toward storing game data in a flexible database. It is still common, however, to use analysis tools for which an appropriate file representation of the data is required as input. I explore the potential for conducting game-theoretic analysis in the database to save the cost of translating the data between representations, as well as to leverage the query optimization and memory management capabilities of the database. Casting the problem in terms of operations that are efficient for

data primarily stored on disk suggests a way of organizing the data that enables efficiently finding pure-strategy Nash equilibria of games with arbitrary role-symmetric representations. Furthermore, when storing game data that is acquired sequentially in a database, we have the potential to conduct interim analysis that may be useful for guiding further sampling, or to amortize the cost of certain analytic operations over the time spent gathering observations. In this vein I discuss tracking which profiles are pure-strategy Nash equilibria, and determining maximal subgames for which we have observations of all profiles.

1.2.4 Bootstrap Methods for Sequential Estimation of Nash Equilibria

Historically, EGTA studies present equilibrium analysis without expressions of statistical confidence. Practitioners employ rules of thumb to determine whether they have acquired enough observations to have confidence in their analysis. This approach not only leaves open the question of whether the analysis should be trusted, as sampling may have terminated with insufficient data to justify the conclusions, but it can also be wasteful, as an ad hoc rule may demand more observations than are required for statistical confidence in the conclusions. A statistical method known as the *bootstrap* has been proposed for conducting statistical analysis on empirical games (Wiedenbeck et al., 2014), and I verify experimentally that this approach remains valid even when data is gathered sequentially. Using the bootstrap to generate confidence intervals, I evaluate the capability of two sampling rules of thumb to return equilibria with low estimated regret confidence bounds. I further propose and evaluate algorithms that use bootstrap-derived confidence intervals as a stopping criterion for sampling sequentially. I find that using that using the bootstrap in this manner enables one to identify δ-Nash equilibria with high confidence, often requiring fewer samples than the rules of thumb.

1.2.5 Application: Equity Premium Estimation in Asset Pricing

One game setting that is inherently large and complex is that of financial markets. In financial research, it is common to model entire markets as one entity that aggregates the preferences and behaviors of the many traders that participate in the market; however, such aggregation removes the impact of specific market microstructure (that is, the details of the environment in which traders interact), and limits the degree to which agent heterogeneity can be included in the model. In contrast, a simulation-based approach can encode arbitrary market microstructure and agent heterogeneity, but can be computationally expensive. As a case study of employing EGTA scaling techniques, I build an empirical game model with a relatively large number of traders to evaluate claims of an analytical pricing model in

simulation. In particular, I examine whether a model of ambiguity aversion can resolve the *equity premium puzzle* (Mehra and Prescott, 1985), the surprising underpricing of equity as compared to risk-free alternatives. I find that the proposed pricing model is not played in Nash equilibria when a natural alternative is available, and that the premium on equity in simulation is low, suggesting that the model does not resolve the puzzle.

1.3 Guide to Reading this Thesis

Some chapters in this thesis assume that the reader has first read earlier chapters. Chapter 2, as an application that predates the development of the scaling methods that are the focus of this work, serves primarily as a motivating example. Readers familiar with EGTA and not especially interested in wireless networking may skip this chapter. Chapter 3 introduces the EGTAOnline experiment management system, which is referenced throughout the rest of the thesis. Chapter 4, in which studying large games in a relational database is considered, builds on the data organization introduced in Chapter 3, but is unrelated to the chapters that follow it. Chapter 5 presents bootstrap algorithms for statistical analysis under iterative EGTA, and is independent of previous chapters, though implementing automated game refinement of this form is discussed in Chapter 3. The equity premium study presented in Chapter 6 was conducted using EGTAOnline (Chapter 3) and includes statistical analysis based on the methods in Chapter 5. The thesis is concluded with Chapter 7, in which I summarize the contributions of the thesis and discuss future avenues for improving the scaling of EGTA.

CHAPTER 2

Application: Wireless Access Point Selection

Consider sitting down at a local coffee shop to do some work on your laptop or tablet. The wireless networking software on your computer informs you that you can connect to the coffee shop's network, the network of the bookstore next door, or perhaps to the public wifi provided by the city. How should you choose which network to use to check your email and browse the web?

Mobile users typically encounter many accessible wireless networks throughout the day, each comprising of one or multiple access points (APs). Each user must decide on the specific set of APs to send traffic to, as emerging technologies support association with multiple APs (Chandra et al., 2004; Shakkottai et al., 2007) simultaneously. To make an informed decision, each user must have some notion of the expected throughput for each AP; however, the most readily available metrics of throughput—signal strength and the network name—may be poor indicators of access point performance, as Nicholson et al. (2006) show. While a user may be able to infer the performance of networks they are using, to obtain performance information about other networks they must resort to probing. These probes—being workloads themselves—impose additional load on the AP, reducing its performance for the incumbent users (Croce et al., 2011). This multi-agent interaction, whereby information gathering behavior impinges on social welfare, suggests modeling the decision of which APs to use as a game.

I construct and analyze empirical game models that capture two emerging wireless technologies: simultaneous association with multiple APs, and using active probing to measure AP performance. Whereas Shakkottai et al. (2007) address the problem of selecting the set of APs to utilize, I focus on modeling AP information gathering, and understanding the impact of probing on the system.[1] To investigate these issues, I consider two models of information gathering: the *bulletin board model* (Kleinberg et al., 2009) in which all players

[1]To the best of my knowledge, this work, originally presented at NetEcon 2011 (Cassell et al., 2011), is the first to investigate the strategic and social welfare implications of active probing on networks.

are periodically alerted of the traffic at all APs, and a model where delay information is distributed only through using or probing an AP.

This investigation is motivated by the question of proposing a protocol for AP selection that can be universally adopted; as such, I model the game as (ex ante) symmetric, and concern myself chiefly with Nash equilibria where all players use the same (possibly mixed) strategy. As such, equilibria correspond to protocols that can be adopted by all clients without incentive to unilaterally deviate to some other strategy. I focus on average delay to users as my measure of solution quality, though there may be other relevant utility attributes.

2.1 Related Work

In the past decade, game-theoretic approaches to the AP selection problem have garnered increased attention. Much of the prior literature has focused on analytical results obtained from modeling AP selection as a load balancing game. Though not expressly addressing AP selection, Suri et al. (2004) proved tight bounds for the *price of anarchy*—the worst-case ratio between the outcomes of selfish play and socially optimal play—in a load balancing game with atomic jobs. Koutsoupias et al. (2007) present similar analysis for the variant where players have only partial knowledge about the delay at a particular AP. I do not expressly address price of anarchy, as calculating the expected delay from equilibrium play in my more complicated, dynamic setting seems intractable, and I cannot even guarantee that I have identified the worst-case equilibrium with the richer strategy space afforded by my model.

Computer scientists have extended load balancing games to more accurately model unique attributes of the AP selection problem. Mittal et al. (2008) consider the ability of wireless users to move physically closer to a less crowded AP to improve signal strength. Cesana et al. (2008) model network selection as a non-cooperative game between users and network service providers. Shakkottai et al. (2007) have explored modeling the problem of multiple simultaneous AP selection as a population game, and examine the costs of running a network under their model. There have even been attempts (Xu et al., 2010) to empirically evaluate strategic AP selection protocols in deployed systems. I focus on the impact of probing on delay incurred by all users, and use EGTA to identify promising AP selection protocols.

2.2 Game Description

I model AP selection as a dynamic load-balancing game with atomic work. The simplest game variant considered has N symmetric players and N identical resources, with each player choosing one AP to process one job, a unit of work of size w, in each period. For consistency with prior literature, I use $w = \frac{1}{N}$ for all simulations. In contrast to one-shot or repeated game models of network congestion, work assigned to an AP in one stage can persist to later stages. Since the link from client to AP is generally of much higher bandwidth than the link from AP to network, the client can send additional requests to the AP before receiving responses to earlier requests. For simplicity, I describe the work processed by an AP per period, k, in terms of jobs.

Players choosing resource a are charged a delay, d_a, equal to the total load on that resource in the current period. Load at an AP can arise from one of three sources: new jobs assigned to the AP in the current period, s_t, jobs assigned to the AP in prior periods that have not been processed by period t, u_t, or probes sent to the AP in the current period, q_t. The delay at AP a in period t, $d_{a,t}$, is then:

$$d_{a,t} = (s_t + u_t)\, w + pq_t,$$

where p is the size of a probe. I first examine a simulation version of the game described by Kleinberg et al. (2009). Players are required to select one resource to send one job to in each period, with information revealed according to the bulletin board model (described in detail below). I then further depart from prior work by explicitly modeling information gathering through probing. For both information settings, I also consider the problem of assigning multiple atomic jobs when APs can be used simultaneously.

2.2.1 Multiple AP Selection

In this variant, in each period, clients have j jobs, where $j > 1$, to assign to available APs. They may choose to use more than one AP, but for each AP beyond the first they are assessed a cost δ for the added complexity of managing connections to multiple APs. If a client uses more than one AP in a period, it is charged the maximum of the delays over these APs. Define $\pi_{c,t}$ to be client c's assignment of its j jobs to available APs in period t, and $A_{c,t}$ the set of APs that are assigned work according to $\pi_{c,t}$. Let $\pi_{-c,t}$ be the assignments made by all other clients in the current period. Client c's objective is then to

minimize its total delay, D_c, over all periods $t \in T$ in the dynamic game:

$$D_c = \sum_{t \in T} \left[(|A_{c,t}| - 1)\delta + \arg\max_{a \in A_{c,t}} d_a(\pi_{c,t}, \pi_{-c,t}) \right].$$

This cost structure defines the benefit of latency hiding provided by switching APs after a request has been sent in order to launch a new request. This capability also carries some risk, as increasing the set of APs that are used increases the likelihood of using an AP with extreme delay.

2.2.2 Information Models

I examine AP selection under two models of information revelation. I begin with the bulletin board model, an information model for load balancing first described by Mitzenmacher (1997). In this model, agents are informed of the delay of each resource at the end of each round. In contrast to the complete information setting, agents are not given the means to perfectly predict how different choices on their part would have affected these delays. This degradation in available information is justified by the difficulties experienced in the real world by clients that attempt to model AP performance. Without knowledge of the number of users of a given AP, distributions of user workloads, and the time it takes for the client's message to be processed at the destination, among other causes of delay, discerning the base capabilities of an AP may be impossible. If the base capabilities are unknown, the client will not be able to predict with any certainty how long its work would have been delayed had they made different choices.

The other information model I consider is one in which a client receives delay information for a particular AP only if the client either sent work to that AP, or sent a probe to the AP to gather this information. Probes have a fixed cost, p, that is incurred by both the probing client and the users of the AP that was probed. Clients are not required to wait for their probes to return, and can still learn some information from ignoring a long-delayed probe. Even without receiving the probe result, the client confirms that the delay at the AP is at least as great as the longest delayed AP that they used in the current period. This probing cost structure exhibits an externality, in that probers are partially insulated from the costs of their probes. If a probed AP has a longer delay than the client's current work assignment, it learns this information at the cost p. The imposed cost to social welfare, however, may be as large as Np. That is, when client c probes a, it adds delay p at a to every user (at most N) of that AP. The additional cost realized by another user i of a may be less than p, if a was not already the most heavily loaded AP in use by i.

2.3 Strategies

Clients adopt strategies that combine a policy for determining an assignment of jobs to APs, and a policy for determining which APs to probe to gather information. To construct the strategy pool, I take the cartesian product of the two policy sets. The following sections detail the base policies for association and probing that I tested, and any variations to them that were necessary to adjust to the different game settings I examined.

2.3.1 Association Policies

2.3.1.1 Random

Perhaps the most obvious strategy for AP selection is to select an AP randomly. If the set of possible actions is selecting a single AP, then for N clients and N identical APs, having all clients randomize over this set of actions is a Nash equilibrium. This equilibrium was identified as the worst case by Koutsoupias and Papadimitriou (2009), as the expected average delay from playing this equilibrium is the furthest from social optimal for the stated action set. When clients have more than one job to schedule per stage, there are two natural extensions of this strategy: choosing one AP at random and sending all jobs to that AP (labeled **R1**), or choosing an AP at random for each job (**RJ**).

2.3.1.2 Hedge Algorithm

The Hedge algorithm is a no-regret online learning algorithm for congestion games. The base Hedge strategy is the bulletin board variant proposed by Kleinberg et al. (2009). The probability of Hedge selecting AP a in some period t is given by:

$$\Pr(a,t) \propto \exp\left(-\varepsilon \sum_{\ell=1}^{t-1} d_{a,\ell}\right),$$

where ε is some small number that governs exploration versus exploitation. For my simulations, $\varepsilon = \frac{1}{v^3\sqrt{t}}$, where v represents the client's belief about the number of users in the system. Though Kleinberg et al.'s analysis requires $v \geq n$, where n is the true number of players, I found that simulation outcomes were not strongly affected by agents over- or underestimating the number of other agents in the system, and present results for the case where $v = n$.

For the multi-job variant of the game, I consider two forms of the Hedge algorithm. **H1** selects a single AP in each period, and sends that AP all its jobs. **HJ** first selects an

AP to send one job according to the Hedge probabilities. All remaining jobs are assigned sequentially according to the following probabilities:

$$\Pr(a,t) \propto \exp\left(-\varepsilon\left[\mathbb{1}_{-a}\delta + s_a w + \sum_{\ell=1}^{t-1} d_{a,\ell}\right]\right),$$

where the indicator $\mathbb{1}_{-a}$ is one if the client has not yet assigned a job to AP a in the current period and zero otherwise, and s_a is the number of jobs that the client has thus far assigned to AP a in the current period. In this way, HJ accounts for the added cost of utilizing more than one AP, as well as the potential benefits of spreading its work over multiple APs. For the probing variant of the game, both forms of Hedge fill in gaps in knowledge about AP delays by assuming an unobserved AP carries delay equal to that of the most recent observation of that AP. Since each client observes a different subset of the APs, differences in selection probabilities can arise in this information model, potentially disrupting convergence to a stable distribution of jobs.

2.3.1.3 Decision-Theoretic Optimization

Given beliefs about the distribution of traffic expected at each AP, clients can optimize their assignment of work while ignoring the choices of other agents. In the single-job setting, this strategy simply chooses the AP that the client believes is least congested prior to the current stage. When the client has to assign multiple jobs per round, there are again two variants: **D1** and **DJ**. D1 sends all jobs to the AP believed to be least congested. In the case of a tie for minimal predicted delay, D1 chooses randomly among the tied APs. DJ sequentially assigns jobs to the AP that has the lowest expected cost, where expected cost of using AP a, c_a, is calculated by:

$$c_a = \bar{d}_a + \mathbb{1}_{-a}\delta + s_a w,$$

where $\bar{d}_a$ is the average observed delay at AP a, and $\mathbb{1}_{-a}$ indicates whether or not the client has already assigned work to a in the current period. In cases where multiple APs have minimal expected cost, DJ chooses randomly among these APs. For the probing variant of the game, D1 and DJ fill gaps in their knowledge in the same manner as H1 and HJ.

2.3.2 Probing Policies

2.3.2.1 Naive Approaches

I consider two naive approaches to probing: probe nothing (**P0**) and probe everything (**PE**). P0 never sends probes to any APs, and functions under the assumption that whatever information is needed for successful decision making can be obtained through trial and error. PE is the other extreme in naive probing; it sends a probe to every AP that the client is not currently using. PE thus replicates the information setting from the bulletin board model, though at significant added cost to the client and to other players.

2.3.2.2 Freshness-Based Probing

If a client's first observation of an AP carries a large delay, the client may never attempt to use that AP, since gaps in knowledge are filled by the most recent observation. To combat this phenomenon, freshness-based probing strategies track how stale are the observations of each AP. **PS** probes the AP that has been observed the least recently, and chooses randomly among ties. If the selected AP corresponds with an AP to which the client is assigning work in the current period, no probes are sent.

2.3.2.3 Variance-Based Probing

If a client's estimate of delay at a particular AP has significantly higher sample variance than other APs, use of that AP carries particular risk, given that client's cost is defined by the maximum among utilized APs. If an AP a carries lower delay than the client expected, the benefit in excess of expectations is bounded by $\arg\max_{a \in A_c} \bar{d}_a - \arg\max_{a \in A_c \setminus a} d_a$, where A_c is the set of APs that the client utilized. In other words, the client realizes this excess benefit from lower than expected delay only if a was expected to carry the highest delay, and this benefit is limited to the difference between the expected delay at a and the actual delay at the most trafficked AP in A_c. In contrast, if the actual delay at a is higher than expected, the added cost is bounded only by constraints of the system such as the number of jobs that have been assigned thus far, and the rate limitations of the APs. Thus clients may be incentivized to take more observations of APs with high sample variance. **PV** probes the AP for which the client's estimate of delay at the AP has the highest sample variance, breaking ties randomly, and ignoring selections that coincide with an AP to which the client is sending traffic.

2.4 Experiments

For each information model, number of jobs for each player to assign in each period j, and AP work clearing rate k, I build an empirical game model through simulation. For all experiments, I use six identical APs and six symmetric players. In each period in simulation, each player is given j constant-size jobs for assignment. I consider two settings for j: $j = 1$ and $j = 5$. Each AP processes k jobs in each stage, with $k \in \{0, 1, 2, 3, 4, 5, 6\}$. After each stage, players receive information about AP delays according to one of the information models described in Section 2.2.2. For the probing model, the cost of a probe was set to be slightly smaller than a job, at 0.125.

Under the bulletin board model, where agents do not need to probe for information, for $j = 1$, the set of strategies considered is {D1, H1, R1}, and for $j = 5$, this set is expanded to include DJ, HJ, and RJ. For the probing model, the set of strategies is given by {D1, H1, R1}×{P0, PE, PS, PV} for $j = 1$, and {D1, DJ, H1, HJ}×{P0, PE, PS, PV} for $j = 5$. R1 and RJ were removed from the association policy set for $j = 5$ as R1 was dominated for all values of k under the probing model with $j = 1$; this reduces the number of profiles to sample from 475,020 to 54,264. For $j = 5$, where agents can use more than one AP at a time, the switching cost δ was set to 0.01—an order of magnitude smaller than the delay of a single job on an empty AP. This difference in magnitude is consistent with state-of-the-art switching implementations (Giustiniano et al., 2009).

To build the empirical game models I constructed payoff estimates by taking the average negative total delay for each strategy in each profile, over 100 observations of the simulation. Each observation consisted of 50 periods of work assignment. To find equilibria of these games, I first performed iterative elimination of dominated strategies to reduce the size of the payoff matrix before solving. After removing dominated strategies, I found approximate symmetric Nash equilibria through *replicator dynamics* (Schuster and Sigmund, 1983), an iterative algorithm that uses the expected payoff of playing pure strategies against a candidate strategy mixture to update the weighting of strategies in the mixture in each step. Friedman (1991) demonstrated that fixed points of this iterative process correspond to symmetric mixed-strategy Nash equilibria with respect to the function used to evaluate strategy performance. Since finite symmetric games are guaranteed to have at least one such equilibrium (Cheng et al., 2004; Nash, 1951), replicator dynamics provides an applicable and straightforward search method. As the outcome of applying replicator dynamics depends upon where the search is started, I seed replicator dynamics with several initial distributions, namely a uniform distribution over the nondominated strategies, and distributions that were weighted heavily towards each one of the nondominated strategies.

j	k	Eq^*	$S_{dominated}$
1	0	H1: 0.71, D1: 0.29	R1
1	1	H1: 0.73, D1: 0.23, R1: 0.04	
1	2	H1: 0.97, D1: 0.02, R1: 0.01	
1	3-6	H1: 1.00	D1, R1
5	0	H1: 0.66, DJ: 0.31, D1: 0.03	HJ, RJ
5	1-3	H1: ≈ 0.65, DJ: ≈ 0.35	HJ, RJ
5	4	H1: 0.67, DJ: 0.32, R1: 0.01	HJ, RJ
5	5	H1: 0.75, DJ: 0.24	
5	6	H1: 0.83, DJ: 0.12, D1: 0.03, R1: 0.02	

Table 2.1: Equilibrium analysis of the bulletin board model.

2.5 Results

With these experiments I address two primary questions:

- What portion of the explored strategy space could form the basis of a widely adopted AP selection protocol?
- How do the new technologies that I model affect social welfare in equilibrium?

I address the first question separately for each information model before tackling the implications of the two models for the second question.

2.5.1 Bulletin Board Model

Table 2.1 summarizes the results of equilibrium analysis of the bulletin board model. In this table, "Eq^*" refers to the equilibrium that has the lowest expected delay among the equilibria found through replicator dynamics, and "$S_{dominated}$" are the strategies that did not survive iterated elimination of dominated strategies. Under the bulletin board model, when players have only one job to schedule in each round and $k \in \{0, 3, 4, 5, 6\}$, R1 is a dominated strategy. This contrasts with previously examined load balancing games, where having all players choose uniformly at random is a Nash equilibrium. R1 is not dominated for $j = 5$, but RJ is dominated for $j = 5$ for most values of k, possibly because it incurs switching costs without explicitly exploiting latency hiding.

Consistent with the work of Kleinberg et al. (2009), I found the Hedge algorithm to be a reasonable strategy for games where players schedule one job each period, even when resources are rate-limited. For $k \in \{3, 4, 5, 6\}$, H1 is actually a pure-strategy Nash equilibrium. The improved performance of Hedge relative to selecting APs uniformly at random

can be attributed to at least two factors. First, in the formulations of Koutsoupias and Papadimitriou (2009) and Kleinberg et al. (2009), the pure strategies comprise the basic actions of assigning work to a particular AP. In my setup, the choice of deviation to a pure strategy is not restricted to picking a single AP to use and committing to it for all rounds, which provides no benefit if all other players are selecting randomly; instead, players have the option of playing strategies that respond to the changing distribution of congestion. Second, players will not want to deviate to selecting an AP uniformly at random from other available strategies when other players are choosing according to a different distribution and the APs are rate-limited. Under this scenario, APs are no longer identical, and thus a strategy that accounts for these changing delays, such as Hedge, can outperform uniformly random selection. Interesting, while H1 performs well, HJ is a dominated strategy for many settings considered. HJ incurs switching costs, yet may not find work assignments that substantially improve on those found with H1.

Decision-theoretic optimization can also be a reasonable approach for these games, provided it is played by a minority of players. In the bulletin board model, all agents have the same information about delays at APs. Agents playing D1 or DJ eventually agree on which AP is expected to have lowest delay and proceed to send their work there. When all agents do so in tandem, the result is the maximum possible delay. These results are similar to the findings of Mitzenmacher (1997) for load balancing in a non-competitive setting when there are significant delays between information updates. Also of note is that the proportion of agents that play D1 or DJ in equilibrium decreases as the clearing rate increases. This is likely because the benefit of playing D1 or DJ over a Hedge strategy is gained by being more thoroughly exploitive of the changing distribution of work in each period, and high clearing rates limit the degree to which APs will differ at the start of each round.

2.5.2 Probing Model

Table 2.2 summarizes the results of equilibrium analysis of the probing model.[2] In contrast to the bulletin board model, most equilibria under the probing model are comprised entirely of decision-theoretic optimization strategies. When $j = 1$, for all but one value of k, D1-P0 is played with probability greater than 99%.[3] Surprisingly, for $k = 1$ D1-P0 is actually dominated, and its place in equilibrium is taken by D1-PS. When APs accumulate

[2]Under the probing model, all strategies that employed either R1 for AP selection or PE for probing were dominated in all experiments, and so are excluded from the table for clarity. HJ is similarly excluded; though undominated in one setting ($j = 5, k = 6$), it is not present in any equilibria.

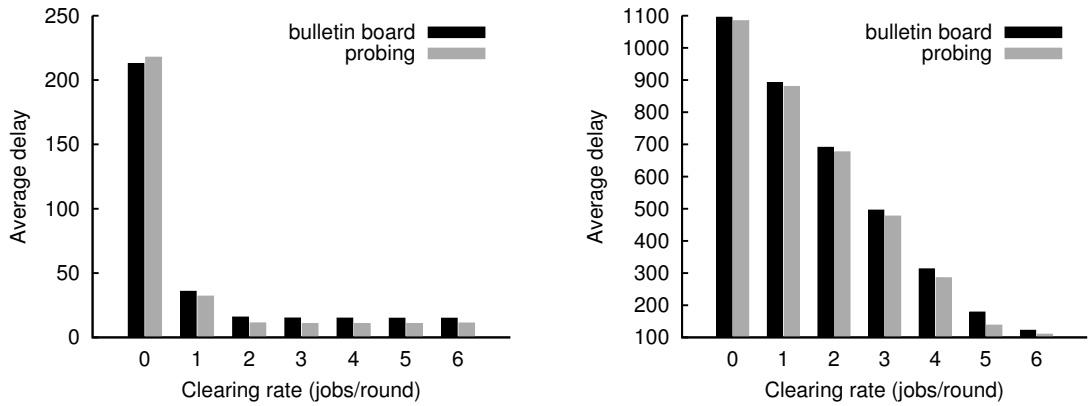
[3]For $j = 1$ and $k = 2$ or 3, H1-P0 is a PSNE, though its expected delay is approximately 30% higher than that of the Eq^* PSNE, D1-P0.

j	k	Eq^*	$S_{dominated}$
1	0	D1-P0: 0.99, H1-P0: 0.01	D1-PS, D1-PV, H1-PS, H1-PV
1	1	D1-PS: 0.88, H1-P0: 0.12	D1-P0, D1-PV, H1-PS, H1-PV
1	2-6	D1-P0: 1.0	D1-PS, D1-PV, H1-PS, H1-PV
5	0	D1-PS: 0.54, D1-P0: 0.46	D1-PV, H1-P0, H1-PS, H1-PV, DJ-P0, DJ-PV
5	1-2	DJ-PS: ≈ 0.99, D1-PS: ≈ 0.01	D1-PV, H1-P0, H1-PS, H1-PV, DJ-P0, DJ-PV
5	3-4	DJ-PS: 1.0	D1-P0, D1-PV, H1-P0, H1-PS, H1-PV, DJ-P0, DJ-PV
5	5	DJ-PS: 1.0	D1-P0, D1-PS, D1-PV, H1-P0, H1-PS, H1-PV, DJ-P0, DJ-PV
5	6	DJ-PS: 0.94, H1-P0: 0.06	H1-PS, H1-PV

Table 2.2: Equilibrium analysis of the probing model.

a significant backlog, it can be worthwhile to probe based on freshness to reduce the risk of players ignoring an AP that has cleared its traffic after previously having been measured as highly congested.

When $j = 5$, for all but one value of k, DJ-PS is played with probability greater than or equal to 94%, including several settings for which DJ-PS is a PSNE. H1 on the other hand, is only found in Eq^* when 6 jobs are cleared from each AP in each period. Thus, whereas Hedge seems to be the most appropriate strategy to use as a protocol under the bulletin board model, decision-theoretic optimization holds that distinction under the probing model. D1 and DJ have dramatically improved performance relative to Hedge strategies under this model, since player beliefs are differentiated by which APs they have observed, and when the observations took place. While this differentiation prevents D1 and DJ agents from making identical assignments, and thus leads to a more even distribution of jobs, it can also prevent Hedge from converging, due to the less predictable landscape.



(a) $j = 1$ (b) $j = 5$

Figure 2.1: Lowest expected delay among found Nash equilibria for different rates of work clearing.

2.5.3 Social Welfare

To address my second question, I compare the lowest expected delays found in equilibrium for each setting considered. This metric is justified particularly in cases where one can designate a focal equilibrium to be implemented, as is the case with a widely adopted protocol. Figure 2.1 compares the two information models under this metric.

One might expect a priori that reducing the amount of freely available information and incurring the externality of probing congestion would dramatically increase expected delay. I find, however, that generally the opposite is true. With exception of $j = 1, k = 0$, players are better off at equilibrium under the probing model than under the bulletin board model. This is because the reduced information actually makes it possible for decision-theoretic optimization to converge to near-optimal assignments. Under the bulletin board model, all players share the same beliefs about the congestion at access points; accordingly, all agents that play the same deterministic strategy will make the same assignment of work, leading to heavy congestion on the access point that was the least congested in the previous period. As such, a randomizing strategy such as Hedge can improves outcomes, though the potential remains to randomly assign work to a heavily congested AP. In contrast, the probing model makes it possible for agents to hold varied beliefs about which access point is the least congested, and this difference in belief leads to more even distributions of work for deterministic association policies. For example, when $j = 5$ and $k = 3$, delay experienced under the optimal assignment of work to APs is 450; playing the PSNE of DJ-PS in this setting results in only a 6% increase, to 477.32, in expected delay. These results suggest that the benefits obtained from probing for information, as opposed to having APs

broadcast their congestion, may outweigh the added social cost that probing imposes.

2.6 Summary: AP Selection Game

Using a recent analytical model as my starting point, I simulated and analyzed three extensions of the load balancing game, for the purpose of modeling the AP selection problem. First, I transformed the load balancing game from a repeated game to a dynamic game, in which work can persist at resources between periods, in order to more accurately capture the overlapping decision making that occurs in AP job assignment. This transformation, combined with the addition of a decision-theoretic policy to the strategy pool, lead to the domination of the uniformly random strategy in most settings. This result contrasts with previous work on the load balancing game, where uniformly random resource selection has been shown to be a Nash equilibrium.

I pursued two further extensions to incorporate the emerging technologies of multiple simultaneous AP use and probing. I found continued support for use of the Hedge algorithm under the bulletin board model; however, under the probing model, Hedge was generally outperformed by decision-theoretic optimization. In the probing model, surprisingly, the reduction in free information, and the significant social cost of gathering additional information through probing, did not result in longer delays in expectation. This result stems from breaking player symmetry within games, mirroring the findings of Mitzenmacher (1997) for non-competitive load balancing. That these results contrast with established game models suggests that we should revisit, and possibly revise, established models when considering the impact of new technologies to strategic decision making.

2.7 Scaling Lessons

The EGTA study presented in this chapter is meant to act as a time capsule and motivating example for the rest of my thesis. Because of the simplicity of the simulation involved, the AP selection game did not necessarily require a huge distributed simulation effort, or major data management system; however, modeling choices for this study were nevertheless influenced by the fact that my simulations were run on a single computer without any significant simulation management tools. The real world scenario that I'm modeling would typically not have symmetric APs, nor would players have identical workloads. Introducing these features, however, would likely increase the variance in observed payoffs, and as I had not yet developed sequential bootstrap sampling methods (Chapter 5), this would have meant taking many more observations to reduce uncertainty in payoff estimates.

At the time this study was originally conducted, I had not yet developed EGTAOnline (Chapter 3), nor did I have any other convenient software tools for searching through profile space in a principled way. Instead, I constrained the strategy space as necessary and gathered observations for complete subgames. As such, I could not afford to examine any parameterizations to the strategies in depth, such as whether to probe more than one stale AP, or to calculate the probabilities for Hedge using a different value for ε. In order to reduce the number of simulations to run for the probing model, I removed some association policies from consideration entirely. Were I to conduct this study today, I would verify that the excluded strategies were not beneficial deviations using one of EGTAOnline's deviation schedulers.

While conducting the study presented here, I also developed the simulator for the equity premium study presented in Chapter 6. It was clear that if I wanted to study a financial market in any significant fidelity I would need a more complex, and thus computationally expensive, simulation model with a greater number of agents than the 6 used for the AP selection game. My desktop computer would not be up to the task this time. Running my simulations in parallel on the university cluster was the next logical step, but would require me learning an esoteric scheduling language, and either writing a large number of scheduling scripts or writing a script to generate scheduling scripts. Recognizing that these challenges were not unique to my simulator, I began work on an automated system for scheduling game simulations to the cluster, culminating in the experiment management system presented in the next chapter, EGTAOnline.

CHAPTER 3

EGTAOnline: Software Infrastructure for Experiment Management

EGTA studies require observing a large space of profiles in simulation, demanding significant computation time. The most direct way to reduce the time required to assemble observation data is to improve sampling throughput by parallelizing simulation. From university-operated clusters to Amazon EC2,[1] researchers increasingly have access to large pools of machines to aid in computational experimentation. Distributed computing is increasingly inexpensive in terms of computation time and monetary cost. Yet human capital costs remain quite high. Researchers wishing to utilize these computational resources typically must learn the technologies for distributing and scheduling the computation, as well as tools for managing the copious amounts of data being created (Sheutz and Harris, 2012). Learning how to leverage distributed computing is often orthogonal to one's research goals, leading to a tradeoff between convenience and limitations on problem scale.

It is hard to quantify how these human capital costs impinge on research production. We can see only what research was produced, not what research might have been conducted had convenient tools been available. Experimenters may unduly limit the scope of studies, for example by capping problem instances at the size tractable for their desktop computers. Such restrictions may detract from the real-world relevance of computational investigations, or otherwise diminish the value of published studies.

The application of EGTA is one form of computational study where restrictions on scale imposed by the use of a single computer may strongly diminish the strength of conclusions. Even when a game is fully symmetric, its profile space stills grows exponentially in the smaller of the number of players and number of strategies. As such the construction of games of even moderate complexity is often beyond the reach of a typical desktop computer. EGTA also carries with it data management concerns due to the substantial amount

[1]The Amazon Elastic Compute Cloud, or EC2, is a service that provides on demand, resizable computation in the cloud. For more information, see http://aws.amazon.com/ec2.

of observation data required. As such, some mechanism for distributing game simulations and retrieving and managing the resulting data is needed.

In my own work, I observed the necessity for such a system when I began the equity premium study presented in Chapter 6. I found myself artificially restricting strategy spaces, and modeling the game with a very coarse-grained hierarchical reduction, in order to construct payoff matrices utilizing only my desktop computer. To alleviate the limitations inherent in using my personal computer for these studies, I created EGTAOnline, an experiment management system designed to make studying large games derived from simulation more convenient through accessible distributed computing and data management. EGTAOnline strives to make the most common aspects of employing the EGTA methodology available through simple web forms, while supporting more complex functionality through a JSON[2] API. Much of the material in this chapter was presented at the AAMAS-12 workshop on Multi-Agent-Based Simulation (Cassell and Wellman, 2013).

Following a review of related efforts, I describe two concepts implemented in EGTAOnline that improve the ability to compactly represent and organize simulation-based game data: *role symmetry* and *data compatibility*. I then detail the EGTAOnline architecture and describe how it supports the application of the EGTA methodology. I also discuss how EGTAOnline supports iterative experimentation through data reuse, and automated refinement of game models through a scheduling API. I conclude this chapter by describing the usage of the system to date.

3.1 Related Work

Many previous efforts have aimed to take advantage of distributed computing for agent-based simulation. One thread of discussion centers on *agent-level parallelism* and how to efficiently distribute a multi-agent based simulation (MABS) over multiple compute nodes. Riley and Riley (2003) present a system for distributed execution of MABS that limits the effect of varying network and system loads on simulation by ensuring that agents are always given sufficient time to think, extending the causal ordering constraints of an earlier parallel and distributed discrete event simulation environment. Mengistu et al. (2008) identify several architectural issues in designing MABS for Grid computing, including threading and communication overhead, and present middleware to address some of these challenges. Alberts et al. (2012) demonstrate that the parallelism afforded by modern graphics cards can be useful for simulations with millions of agents, as may be necessary when simulat-

[2]JSON stands for the Javascript Object Notation, and is a common file interchange format akin to XML. For more information, see `http://www.json.org`.

ing biological systems. In contrast, *simulation-level parallelism*, as employed for example by Bononi et al. (2005), distributes simulation runs, possibly with differing run-time parameters, across multiple compute nodes. EGTAOnline likewise applies parallelism at the simulation level, and exploits the flexibility of specifying different run-time parameters to simulate multiple strategy profiles in parallel. Game simulation is particularly amenable to the exploitation of simulation-level parallelism, as observations of different profiles are independent and the number of profiles to observe is tremendous.

EGTAOnline builds on a tradition of tool-building in the computational game theory community. McKelvey et al. (2006) describe Gambit, a collection of game-specification tools and analysis algorithms. GAMUT (Nudelman et al., 2004) offers functions for generating random instances from an extensive set of game classes. Both of these toolkits support analytically specified games, whereas EGTAOnline is built to address the construction of games from simulation data. EGTAOnline was also inspired by two existing systems, developed by Jordan et al. (2007) and Collins et al. (2009) respectively, which provided web interfaces for scheduling simulations of the TAC Supply Chain Management game (TAC SCM).

3.2 Role Symmetry

As discussed in Chapter 1, when games are fully symmetric, they can be represented more compactly since the payoff for playing a particular strategy in a profile depends only on the number of players playing each strategy. Some games of interest are not fully symmetric in this way, yet the players can be partitioned into groups such that players are symmetric within the group. Consider, for example, a market in which buyers are symmetric and sellers are symmetric, but the strategies and payoff functions for the two groups are different. In such cases, we may still exploit this symmetry within roles through a *role-symmetric representation* of the game.

A role-symmetric representation partitions the set of players in a game into some number of disjoint sets that I term *roles*, within which players share a strategy set and are symmetric with respect to payoffs. Furthermore, the payoff to players outside of role r may depend only on the counts of strategies assigned to players in r, not on the identity of players playing a particular strategy. A role-symmetric representation of a game is given by $\Gamma = \langle \{I_r\}, \{S_r\}, u(\cdot) \rangle$, where $\{I_r\}$ is a partition of players into roles, and S_r is the strategy set of players in role r. The utility function $u(\cdot)$ is restricted such that if two players in the same role swap strategies, their entries in the payoff vector are swapped, and all other payoffs remain unchanged. For the forthcoming discussion, let $R = \{r\}$ be the index set

of roles, $N_r = |I_r|$ be the number of players in role r, and $M_r = |S_r|$ be the size of the strategy set available to players in r.

In a role-symmetric representation, the set of distinct assignments for players in a single role is the same as for a fully symmetric game with number of players equal to the size of the role and strategy set equal to the role's strategy set: for the role r with size N_r and strategy set of size M_r, there are $\binom{N_r+M_r-1}{N_r}$ distinct assignments of strategies to players in the role. We can think of assigning the strategy counts to a particular role as analogous to the choice of strategy for a single player in a fully non-symmetric game. As such, the number of distinct assignments in the full game is given by

$$\prod_{r \in R} \binom{N_r + M_r - 1}{N_r}.$$

This representation generalizes fully symmetric (games with a single role) and fully non-symmetric games (games in which each player has its own role). Game data in EGTAOnline uses this representation due to its generality and compactness. By supporting symmetry by roles in EGTAOnline, users of EGTAOnline are now able to conduct simulation studies that exploit an intermediate degree of symmetry where appropriate. For example, Wah and Wellman (2014) examine the impact of a market maker on background traders, treating the two classifications of traders as separate roles. Prior to EGTAOnline, such a study would typically have the strategy of one of the roles specified exogenously, or treat all players as symmetric and suffer from a model that did not match the underlying game.

3.3 Data Compatibility

EGTAOnline allows one to construct games from a database of simulation results. Following Vorobeychik and Wellman (2009)'s description of simulation-based games, EGTAOnline expects a simulator to act as an oracle $\mathcal{O}$ for some underlying game with utility function $u(\cdot)$, returning sample payoff observations such that, for any profile s, $\mathbb{E}[\mathcal{O}(s)] = u(s)$. In other words, a simulator can function as an oracle for some underlying game if the expected payoffs to each player in simulation are consistent with the utility function for the game being simulated.

Where I extend this description of game simulation in EGTAOnline is through a more explicit description of what makes data compatible—able to be considered together as an empirical game for game-theoretic analysis. This extension is needed since EGTAOnline stores data for multiple experiments in a single database. In particular, several different

experiments may be conducted with the same simulator program, yielding data that is collectively incompatible for the purposes of game-theoretic analysis. The oracle model in EGTAOnline parameterizes the simulator program $\mathcal{O}$ with a run-time configuration c, resulting in a *simulator instance* $\mathcal{O}_c$. Two pieces of data are compatible only if they were generated by the same simulator instance. Furthermore, since role membership is not a strategic parameter for players, for data of two profiles to be compatible they must have the same partition of players into roles. As such, a formal description of a game in EGTAOnline can be given by

$$\Gamma = \langle \{I_r\}, \{S_r\}, \mathcal{O}_c, \Theta \rangle ,$$

where Θ is the set of observations of $\mathcal{O}_c$ for the profile space induced by $\{I_r\}$ and $\{S_r\}$.

3.4 Architecture

EGTAOnline provides empirical game analysts with distributed simulation scheduling and a robust data storage solution. Users of EGTAOnline can take advantage of the parallel and distributed computation afforded by a large cluster without having to learn the details of scheduling jobs onto the cluster. Users also benefit from a database management system for storing observation data without having to learn a database query language. As such, barriers to constructing large simulation-based games are dramatically reduced.

Figure 3.1 illustrates the role of EGTAOnline in the iterative EGTA process. The following subsections present the primary conceptual entities of EGTAOnline and explain how they support the construction of empirical games.

3.4.1 Simulators

To use EGTAOnline, users must write a simulator program, which takes as input a run-time configuration c and a pure-strategy profile s to sample, and outputs an observation θ. The exchange of simulator input and output is conducted through a simple, file-based protocol, accommodating simulators developed with any programming language or simulation platform. As noted in Section 3.3, multiple simulator instances can be derived from a single simulator program through the specification of run-time parameters. As such, a user can upload a single simulator program and perform multiple independent experiments by specifying different configurations of run-time parameters, as well as varying the number of players and partitions of players into roles, in the scheduler interface, described in Section 3.4.3.

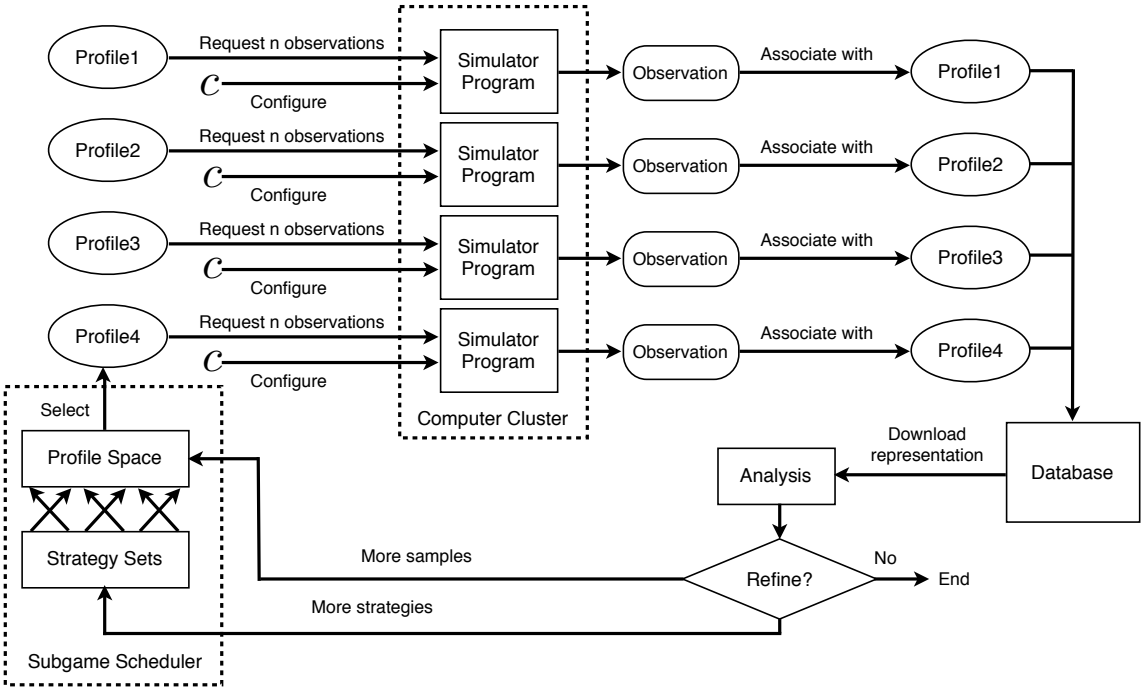


Figure 3.1: Supporting the EGTA process with EGTAOnline.

3.4.2 Observations

An *observation object* includes the vector of payoffs observed in simulation and, optionally, additional statistics about the simulation. These optional simulation statistics can be used in variance reduction techniques (L'Ecuyer, 1994) to improve payoff estimates. One form of variance reduction that is explicitly supported in EGTAOnline is the method of *control variates* (Lavenberg and Welch, 1981). This method is based upon reducing variance in simulation outcomes by exploiting knowledge of how outcomes covary with the realization of random variables known as control variables. Having recorded realizations of their chosen control variables during simulation, users of EGTAOnline may specify adjustment coefficients for applying control variate adjustments to payoff estimates through a web form. Of note is that, since payoffs for different roles may differ greatly in scale, and may covary differently with the control variables, users must specify their adjustment coefficients on a per-role basis. Additionally, users have the flexibility to specify control variables at the level of individual players, such as players' private values, and to use different sets of player control variables for different roles.

Secondary simulation statistics may also record the outcome of variables tied to agent interaction, enabling analysis of the impact of strategic choices on non-payoff variables. I exploit this capability to reason about pricing behavior at strategic equilibrium in Chapter 6.

3.4.3 Schedulers

Once a simulator has been registered with EGTAOnline, the experimenter may create one or more *schedulers* for that simulator. Schedulers provide a convenient way for users to specify a large quantity of simulations to run on a cluster. Specifically, schedulers take as input:

- running requirements, such as memory and time, that the simulator needs to take an observation,
- sampling information, such as maximum number of observations to gather per profile, and number of observations to gather in a single job on the cluster,[3]
- the configuration c of run-time parameters to use with the simulator program, and
- $\{s\}$, the collection of profiles to sample.

It is generally inconvenient to specify the set of profiles to sample through direct enumeration. EGTAOnline therefore provides facilities to define combinations of profiles generated according to a specified pattern. The current implementation supports schedulers based on three particularly useful patterns.

The first, *subgame*, generates profiles defining a subgame by specifying a partition of players into roles $\{I_r\}$, and restricted strategy sets $S'_r \subseteq S_r$ for each role r. Subgame schedulers construct the profile space associated with $\{I_r\}$ and $\{S'_r\}$ as described in Section 3.2. For example, consider a subgame scheduler with $I = \{\{i, j\}, \{k\}\}$ and $S' = \{\{s_1\}, \{s_2, s_3\}\}$. To create the space of profiles to sample, the scheduler first computes the set of possible assignments to each role, respecting symmetry, resulting in $\{\{s_1, s_1\}\}$ and $\{\{s_2\}, \{s_3\}\}$. The scheduler then takes the cross product of these sets to generate the set of profile strategy assignments, $\{(\{s_1, s_1\}, \{s_2\}), (\{s_1, s_1\}, \{s_3\})\}$. This scheduler would thus sample for the two profiles with assignments $(\{s_1, s_1\}, \{s_2\})$ and $(\{s_1, s_1\}, \{s_3\})$.

The *deviation* pattern expands on a base set of profiles generated by a subgame scheduler by considering single-player deviations to alternative strategies. Users specify a partition of players into roles, and for each role r, two disjoint, restricted strategy sets: the base set S'_r and deviation set S''_r. The deviation scheduler uses the base sets $\{S'_r\}$ to generate profiles defining the full subgame over these strategies, as described above. To the subgame induced by $\{I_r\}$ and $\{S'_r\}$ are added any profiles that can be reached through

[3]Breaking the requested observations across multiple jobs on the cluster allows users to get some data for every profile early in the process, enabling game-theoretic analysis before all requested observations have been taken.

one player switching to a strategy in its role's deviating strategy set, S''_r. Consider our example from before, with $I = \{\{i, j\}, \{k\}\}$ and $S' = \{\{s_1\}, \{s_2, s_3\}\}$, but instead of a subgame scheduler we use a deviation scheduler and specify $S'' = \{\{s_4\}, \{s_5\}\}$. In addition to the profiles in the subgame, $\{(\{s_1, s_1\}, \{s_2\}), (\{s_1, s_1\}, \{s_3\})\}$, this scheduler would also sample from $\{(\{s_1, s_4\}, \{s_2\}), (\{s_1, s_4\}, \{s_3\}), (\{s_1, s_1\}, \{s_5\})\}$. The deviation scheduler supports incrementally searching for payoff-improving strategy deviations, without constructing the exponentially larger subgame induced by adding these strategies to the base strategy sets, enabling iterative strategy exploration (Jordan et al., 2010).

The final scheduling pattern, *reduction*, generates profiles defining subgames (and optionally, deviations) for approximations based on reducing the effective number of players in the game. EGTAOnline supports both types of reduction mentioned in Chapter 1: hierarchical reduction and deviation-preserving reduction. Though profiles for either reduction scheduler are selected from a game with a reduced number of players, the profile objects that are stored in the database represent the assignment of strategies to agents in the unreduced game. For example, when a 2-player hierarchical reduction of a 4-player symmetric game requires a profile where one player, controlling two agents, plays strategy s_1 and the other player, also controlling two agents, plays strategy s_2, the profile object that is requested of the simulator and stored in the database is (s_1, s_1, s_2, s_2). Consequently, observations gathered under a hierarchical reduction scheduler may also be used in more fine-grained reduced game models, as well as in an unreduced game model. This feature is discussed in greater depth in Section 3.5.

When constructing a reduction for games with multiple roles, different roles may reduce the number of players in different proportions. This feature can be useful when the strategy choices of players in a specific role have a greater impact on outcomes than the choices of players in other roles. If we were modeling the mortgage market, for example, we may assume that changes in lending strategy by banks have a greater impact on outcomes than changes in the borrowing strategy adopted by individual home buyers, and thus want to approximate the banks' strategic choices more precisely than those of borrowers. Furthermore, any number of full-game players can be abstracted to any smaller number of reduced-game players by scheduling profiles for which the desired ratio between full-game players and reduced-game players holds approximately. For example, scheduling for a symmetric game with a hierarchical reduction from 5 players to 2 players may simulate the requested profile (s_1, s_2) as $(s_1, s_1, s_1, s_2, s_2)$, as the desired ratio of 2.5 full-game players for every reduced game player can only be met approximately.

To enable arbitrary profile sampling behavior, EGTAOnline allows users to specify *generic schedulers*. Profiles to sample, and the number of observations requested, are

passed to these schedulers through a JSON API. Users can write scripts with complex logic determining which profiles to sample, sending requests to EGTAOnline to update the scheduler accordingly. This feature, combined with the ability to download games via the JSON API, provides the flexibility necessary to support automated refinement of game models, discussed further in Section 3.6.

3.4.4 Simulations

A *simulation object* in EGTAOnline summarizes the state of a simulation job that has been scheduled on the cluster. A simulation job requests observations be taken for a single profile using a specified simulator instance, typically bundling together more than one observation so as to amortize the overhead of scheduling on the cluster. Simulation objects record the status of a job and any associated error messages, with access to this information provided through the EGTAOnline web page. The status of a job may be any of the following:

pending the job has not yet been scheduled to the cluster

queued the job is in a job queue on the cluster and is scheduled to run when resources become available

running the job is running on the cluster

processing the job has finished running on the cluster and its data is being processed by EGTAOnline

failed the job has reached an error state and is no longer running on the cluster

Errors can be caused by system problems, such as loss of network connectivity, failures in running the simulator, or any programmer-defined error. When a simulation returns with an error, the data gathered for that simulation is marked as invalid. Simulator programmers are encouraged to supply an informative error message whenever a state is reached that invalidates the observation data. This allows the user to detect and address error states that might be too rare to show up in preliminary testing, but manifest themselves when many observations are gathered. Diagnosing error states is important as consistent errors can lead to sample bias. For example, if certain realizations of random variables always crash the simulator, then the data set that you are able to collect will not accurately correspond with the specified distribution of random variables.

3.4.5 Profiles

An EGTAOnline *profile object* associates a collection of observation objects with the pure-strategy profile and simulator instance $\mathcal{O}_c$ that generated those observations. As discussed in Section 3.3, distinguishing profile objects by simulator instance enables consistent storage and use of observation data from many experiments, which may have overlapping profile spaces. When a profile object already exists for a given simulator instance and strategy assignment, any new data gathered is associated with that profile object. Thus, profile objects can be in the profile set of multiple schedulers, and associated with multiple game objects, allowing observational data to be included in all relevant analysis contexts—a topic I revisit in Section 3.5.

3.4.6 Games

A *game object* provides filtered views onto the available data, corresponding with a specified game. A game object defines a collection of compatible data based on a simulator instance, partition of players into roles, and a strategy set for each role. When users request a representation of the game object, profile objects that match the specified criteria are collected and rendered for the user at one of three available levels of detail. The available levels of detail are:

summary contains the sample mean and standard deviation of observed payoffs for each distinct strategy in each profile.

observations includes the observations of each profile, with simulation-level feature information and players summarized according to symmetry, giving the sample mean and standard deviation of payoffs within the observation for each set of players that played the same strategy.

full includes observations of each profile in full detail, with simulation-level feature information and distinct payoff and feature information for every player.

3.5 Data Reuse

Gathering simulation data is a costly enterprise, particularly when many thousand different scenarios must be simulated as in the construction of some empirical games. As such, we would like to maximize the value of previously gathered data through extensive reuse. Data reuse is a natural consequence of the iterative EGTA process (Figures 1.1 and 3.1),

as game analysis and refinement decisions are made on an ever-expanding set of observations. This aspect of EGTA contrasts with many other applications of MABS. MABS studies typically observe fixed, but potentially adaptive, agent behavior in a particular simulated scenario. Although such studies may examine several different scenarios through a parameter sweep, the data from different scenarios are not analyzed together. Game-theoretic analysis, however, is based on comparing the outcomes of scenarios that differ by agent strategy selection.

Through the use of game objects, EGTAOnline makes it easy to compare observations in multiple relevant game-theoretic contexts. Consider two game objects differing only in their strategy sets. By the definition provided in Section 3.3, the data for both games are mutually compatible. Thus, if both game objects specify a single role with a strategy set that includes strategy A, then observations of the profile where all players play A will be present in both game objects. Similarly, if we create a third game object that has as its strategy set the union of the strategies present in the first two game objects, its representation will have all of the observations present in the other two game objects. Even though this larger game object subsumes the data from the other two, it is not always the preferred view of the data. For example, in Wah and Wellman (2014)'s market maker study, they observe how traders behave in equilibrium without a market maker by restricting the market maker to a "no-op" strategy.[4] In this example, restricting the strategy set of the market maker in this way leads to a qualitatively different game. Furthermore, since most game analysis is super-linear in the number of profiles, constructing the game corresponding to only the profiles currently under consideration allows EGTAOnline to spend less time rendering the game's representation, and users to spend less time downloading and analyzing it.

As EGTAOnline provides a persistent data store, it is also easy to reexamine experiments long after they were originally conducted. If a new strategy is proposed for a particular scenario, testing whether it disrupts previous findings leverages all the previously gathered observations. Using a deviation scheduler, we can select profiles that correspond to unilateral deviations to the new strategy and measure whether such deviations lead to regret in previous equilibrium candidates. If the new strategy is a beneficial deviation, subsequent exploration still benefits from previously gathered observations as the profile space induced by adding this strategy to players' strategy sets contains the space of previously sampled profiles.

Two procedures that I cover in the equity premium study in Chapter 6 benefit directly from the data reuse supported by EGTAOnline. In Chapter 6, I describe how to estimate the

[4]The way in which Wah and Wellman build games without a market maker is not described in their text. This information comes from conversations with the authors.

expectation of non-payoff variables by weighting observations according to the probability that their associated profile is played in equilibrium. More generally, any solution concept can be used to calculate a conditional expectation estimate. EGTA typically employs Nash equilibrium as its primary solution concept, but we may be interested in how a variable behaves if, for example, we assume all players play a single strategy, or players' choices of strategy are correlated. As with the addition of a new strategy, if a new solution concept is proposed we can use all of our previously gathered observations in constructing the new estimate.

The second procedure that benefits from data reuse is the Hierarchical-Reduction-Based Search (HRBS), an equilibrium search method based upon hierarchical reductions of increasing granularity, detailed more extensively in Chapter 6. As mentioned above, when employing a reduction scheduler observations are associated with profiles over the unreduced set of players. To analyze a reduced-game model, the user renders a representation of a game object with the unreduced player set and transforms the resulting observation matrix as specified by the reduction. For a 2-player reduction of a 4-player symmetric game with the strategy set $\{A, B\}$, for example, this means selecting the profiles $\{(A, A, A, A), (A, A, B, B), (B, B, B, B)\}$ and treating them as though they were the profiles $\{(A, A), (A, B), (B, B)\}$. A consequence of this selection procedure is that hierarchical reductions of different sizes may select overlapping sets of unreduced profiles. In the HRBS, more fine-grained reduction schedulers often reuse many observations taken under coarse-grained reductions as the overlap in unreduced profile space can be significant.

Using a stylized version of HRBS, I can demonstrate how significant this data reuse can be. Assume a fully symmetric game and the following scheduling procedure: for each divisor n of N, the number of players in the unreduced game, in increasing order, we sample the space of profiles specified by the n-player reduction with strategy set of size M. The number of previously unobserved profiles specified by the n-player reduction step is thus given by the recursive relation:

$$f(n, M) = \binom{n + M - 1}{n} - \sum_{n'|n} f(n', M).$$

This relation follows from recognizing that when n' divides n, every profile in the n'-player reduction corresponds to a profile in the n-player reduction. Figure 3.2 demonstrates the fraction of profiles that are covered by smaller reductions under this procedure, $\sum_{n'|n} f(n', M)/\binom{n+M-1}{n}$, for selected values of n and M. We can see that this fraction depends on the number of strategies as well as the divisors of n, but not explicitly on N, the number of players in the unreduced game. If we restrict our attention to n that are prime,

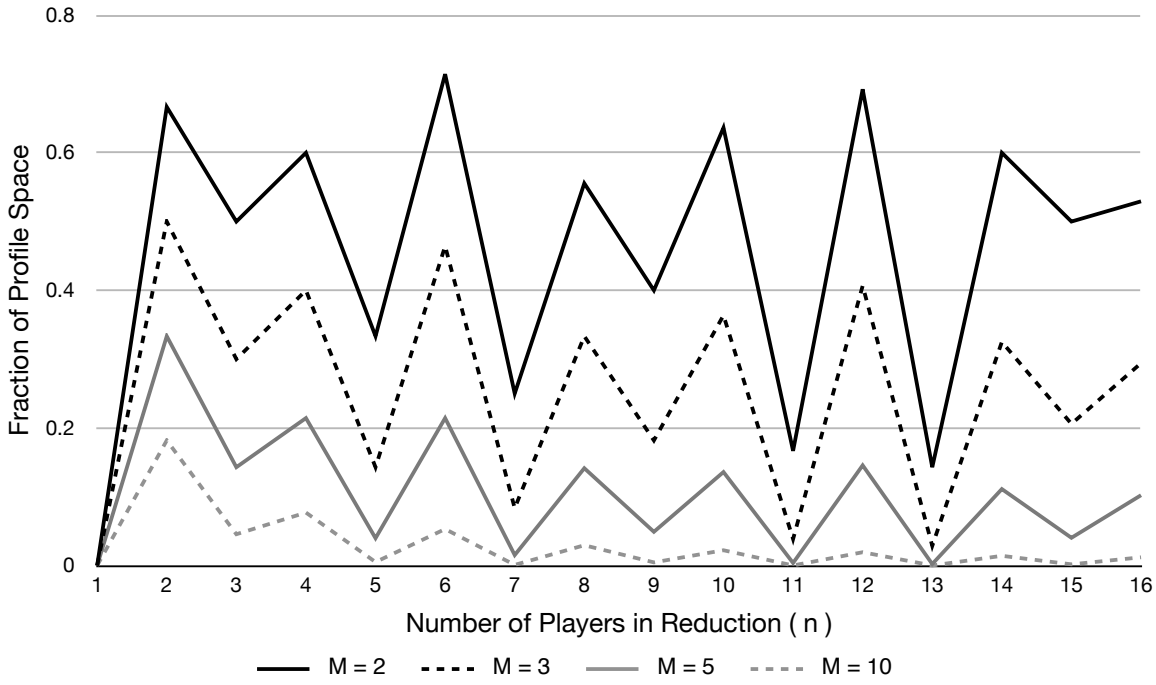


Figure 3.2: Fraction of unreduced-game profile space for an n-player reduction that is covered by smaller reductions, shown for varying size of strategy set, M.

this value decreases as larger n are selected, whereas the relationship is non-monotonic for composite values of n. For small values of M, the prospect of performing an additional iteration of the equilibrium search is much less daunting, since 40–60% of the space may have been explored in earlier steps. This level of data reuse between steps increases the likelihood that the equilibrium candidates that we identify in each step are close to those identified in previous steps, though it is unclear whether this property is beneficial; when the candidates are not approximate equilibria, we may proceed through several iterations of the search before sufficient evidence is uncovered to refute them as candidates.

3.6 Automated Refinement of Game Models

Typically, the game refinement step of the EGTA methodology requires human intervention. An empirical game analyst defines an experiment, performs the required simulation, and analyzes the resulting game model. At this point, they either report findings or set up another experiment, repeating the previous steps. These decisions could be made algorithmically, especially when future experiments are uniquely determined by the outcome of analysis. Practically though, interacting with EGTAOnline through submitting web forms is not optimized for computer-to-computer interaction.

To make automating the game refinement step simpler, EGTAOnline provides API access to its basic control functions. This API allows users to construct complex scripts that interact with EGTAOnline directly, bypassing the web interface. I describe two applications of automated game refinement, including the sequential estimation of Nash equilibria (discussed in detail in Chapter 5), and how they can be implemented with EGTAOnline.

3.6.1 Exploration of Profile Space

Jordan et al. (2008) examined several algorithms to tackle the problem of exploring a game's profile space to quickly identify a Nash equilibrium. The authors treat identifying a Nash equilibrium as a search problem where each step identifies the next profile to sample. These algorithms are designed to sample profiles sequentially, focusing on identifying the *single best* profile to sample at any point in time. With EGTAOnline, several profiles may be sampled in parallel with little added cost. As such, extra information can be gathered in every step, and individual profile selection may be suboptimal.

One profile selection algorithm proposed by Jordan et al. is Minimum-Regret-First Search (MRFS), which uses estimates of regret to guide search. The key concept behind MRFS is that for every profile s, at any step in the search, there is an estimated lower bound on the regret of s, $\hat{\epsilon}(s)$, defined to be the maximum payoff improvement thus far observed from evaluating profiles in $\mathcal{D}(s)$, the deviation set of s. Once all profiles in $\mathcal{D}(s)$ have been evaluated, the value of $\epsilon\,(s)$ is *confirmed*. If the confirmed regret of a profile is zero, it is a Nash equilibrium.

At each step, MRFS chooses to sample a previously unobserved deviation from the profile s with the lowest unconfirmed regret bound. The profile to sample, $\bar{s}$, is chosen with the function SELECT-DEVIATION, which attempts to predict which profile is likely to provide the greatest benefit to the deviating player. After $\bar{s}$ has been sampled, the regret bounds of $\bar{s}$ and all profiles in $\mathcal{D}(\bar{s})$ are updated to reflect this new data.

Algorithm 3.1 presents a modification of MRFS to take advantage of parallel profile sampling. The Minimum-Regret-First Search with Parallel Sampling (MRFSPS) schedules k profiles to be sampled in every step. It achieves this by replacing SELECT-DEVIATION with SELECT-MULTI-DEVIATIONS. When the profile s has more than k unobserved deviating profiles, the k deviating profiles most likely to increase $\hat{\epsilon}(s)$ are chosen for sampling. If the target profile s has no more than k unobserved deviating profiles, all deviating profiles are selected, and the profile with the next lowest unconfirmed regret is considered. The algorithm continues in this manner until k profiles have been selected for sampling, scheduling them to be sampled in parallel.

Algorithm 3.1 Minimum-Regret-First Search with Parallel Sampling

```
Select first profile to sample at random, and add this profile to Queue
while Queue is not empty do
    ℓ ← ∅
    P ← ∅
    while |P| < k and ℓ does not contain all profiles in Queue do
        Select from Queue the lowest ε̂(s) profile s not already in ℓ
        if s is confirmed then
            Remove s from Queue
            ε(s) ← ε̂(s)
        else
            ℓ ← ℓ ∪ {s}
            P ← P ∪ SELECT-MULTI-DEVIATIONS(s, k − |P|)
        end if
    end while
    Sample all s̄ ∈ P in parallel
    for s̄ ∈ P do
        Insert s̄ into Queue if previously unevaluated
        Update ε̂(ŝ) for ŝ ∈ {s̄} ∪ D(s̄) in Queue
    end for
end while
```

This algorithm is just one of several possible modifications to MRFS to take advantage of parallel sampling. Other algorithms discussed by Jordan et al. (2008), can also be modified to benefit from this capability. Recently, Wellman et al. (2013) presented an algorithm, implemented as a control script for EGTAOnline using the JSON API, that automates strategy exploration. Though their algorithm does not reason about parallelism explicitly, it benefits from parallelism by requesting a large number of profiles at once, either to complete a subgame or to evaluate potential deviation strategies.

3.6.2 Sequential Estimation of Empirical Games

Analyzing simulation-based games presents an added challenge over analytically specified games. Given the stochastic nature of simulation, how does one ensure that equilibria identified for an empirical game model are good approximations of equilibria of the game described by the simulator? In Chapter 5, I address this question by presenting algorithms for sequentially sampling until δ-Nash equilibria are identified with statistical confidence. In my experiments, however, I do not sample from a simulator directly, but instead use either simple, randomly generated games, or draw from an existing set of simulation observations. One may wonder how difficult it is to implement these algorithms for practical

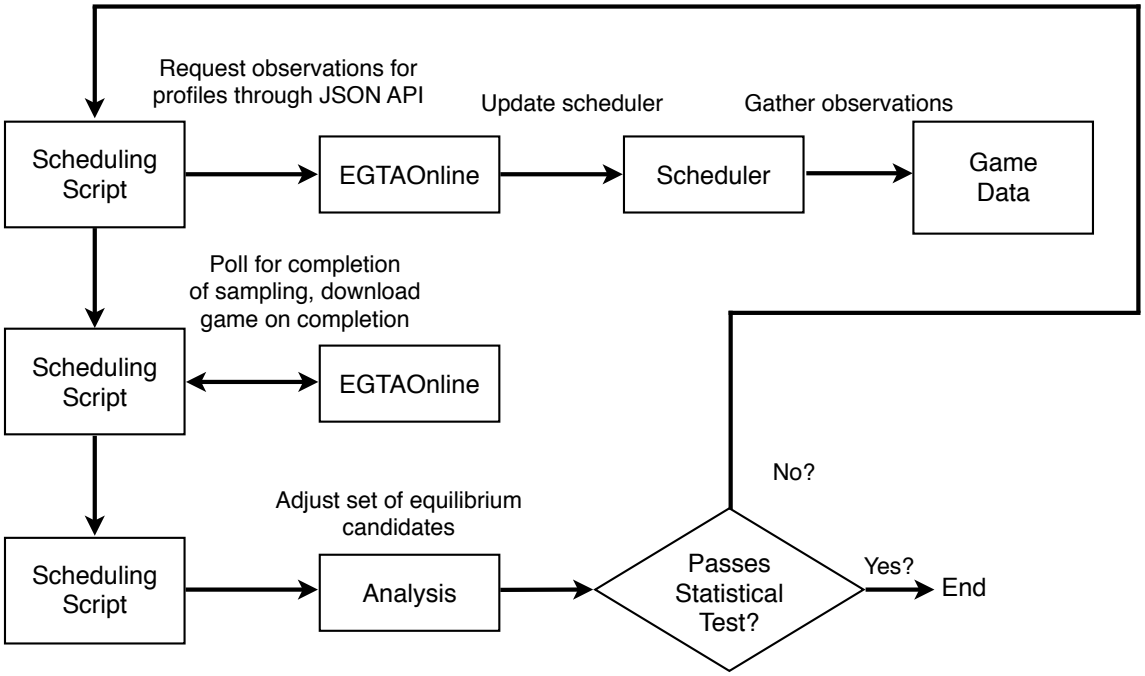


Figure 3.3: Implementing a sequential estimation procedure with EGTAOnline.

use.

Figure 3.3 demonstrates how to conduct such sequential sampling procedures using EGTAOnline.[5] The procedure begins by requesting an initial set of profiles and number of observations to gather for each profile. The algorithms described in Chapter 5 request the same number of observations for each profile, but EGTAOnline provides the flexibility to request different numbers of observations for each profile. Periodically the procedure must poll EGTAOnline to determine if the requested observations have been gathered. Once the procedure observes that the current batch of sampling is complete, it renders a representation of the game from EGTAOnline. Next the procedure analyzes the game to determine if some stopping criteria have been met. In the case of the algorithms described in Chapter 5, this involves first computing equilibria of the current game model, then evaluating the available statistical evidence. The procedure continues in this fashion, requesting further observations until the stopping criteria is met.

[5]Not shown on this diagram is the work to start the cycle, but this is not particularly onerous. The empirical game analyst first registers a generic scheduler for the simulator instance of interest. The analyst then supplies their scheduling script with the scheduler's identifier and their authentication information for connecting to EGTAOnline, as well any other parameters required by their scheduling algorithm.

3.7 In Production

I developed EGTAOnline to address a perceived need for robust sampling infrastructure to support the EGTA methodology. One way to assess if I have achieved this goal is to observe how the system is used by practitioners of the methodology, and how heavily they use the system. Over the last three years of use, approximately 27 million observations were recorded for $560,000$ profiles, totaling over one billion payoff values. Many of these observations were generated for experiments detailed by Wellman et al. (2012), Dandekar et al. (2012), Wellman et al. (2013), and Wah and Wellman (2014), as well as the equity premium study presented in Chapter 6. Each of these studies can be said to focus on a large game, as the smallest of these games has a full profile space of nearly $600,000$ profiles.

Our database currently has 14 distinct simulators registered, with multiple versions of some of these. Schedulers (297) and games (129) significantly outnumber registered simulators (34), and have been used for exploring different simulator configurations and different profile spaces. Users regularly modify or delete schedulers and game objects, as doing so does not affect stored observations, making these numbers significant underestimates of the number of the experiments that have been carried out thus far. There is considerable variety among the experiments conducted so far, ranging from explorations of TAC SCM, where the simulation requires the parallel cooperation of multiple nodes, to an introduction-based routing protocol (IBRP) (Wellman et al., 2013), where role symmetry and reduction are exploited. The IBRP study also used the JSON API to automate exploration of a space of parameterizeable strategies. Thus far EGTAOnline has received considerable use in our research lab, use which should become even more efficient, and thus enable even more extensive use, with the implementation of the techniques discussed in the next two chapters.

CHAPTER 4

Efficient Analysis of Large Game Data Sets

With EGTAOnline, we can build games from vast amounts of simulation data. Although there has been considerable work on algorithmic issues in game theory (Nisan et al., 2007), little research effort has focused on scaling the data-centric elements of game-theoretic modeling. The way in which game data sets are stored and accessed may have a marked impact on the costs of analysis. A payoff matrix, for example, must be reduced from N dimensions to 1 when stored on disk or in main memory. This removes the locality of deviation sets found in the payoff matrix, where deviations for a particular player are reached by traversing a single dimension. If algorithms are not designed with the computer representation of a game in mind, they may access the data in a way that leads to increased costs from moving data back and forth between cache and memory, and memory and disk.

Another concern in analyzing large game data sets is the cost of translating the stored game representation into a model suitable for computation. Gambit (McKelvey et al., 2006), a prominent software toolkit for game-theoretic analysis, for example, takes as input a file representation of a game and builds an in-memory object representation on which it conducts analysis. The cost of building an object representation of a game at best grows linearly with the size of the file representation, and may grow superlinearly if building the object representation requires any non-trivial processing, such as maintaining a specific ordering of the data. As I show in this chapter, this translation step can be the dominant cost for some analysis tasks, regardless of the size of the input.

In this chapter I discuss the storage and analysis of games within a relational database system. By moving analysis closer to the data, I aim to reduce translation costs, as well as reduce costs associated with poor data-access patterns. After presenting some background and terminology, I introduce a novel schema for storing game data that exploits the role-symmetric representation of games to store games compactly, and construct SQL implementations for primitive operations common to game-theoretic analysis.[1] I compare the

[1]This material was presented in a preliminary report at the SIGMOD-14 Data Science for Macro-Modeling workshop (Cassell and Wellman, 2014).

performance of a SQL query for the set of pure-strategy Nash equilibria (PSNE) in a game to two PSNE enumeration algorithms implemented in Gambit. I also present algorithms for identifying *maximal complete subgames*—maximal strategy sets for which all profiles have been observed—by updating completeness information as observations are added to the database.

4.1 Background: Database Management

In this chapter I focus on implementing game-theoretic concepts within a *relational database management system* (RDBMS). In a relational database, data is stored in tables with a fixed layout, or schema, with each table corresponding to a different type of entity in the system, and each row in a table providing data for a single entity. A relation is a combination of data from one or more tables according to some specified rules (Codd, 1970). The relational model allows for storing related entities in separate tables so as to limit data dependencies among entities of different types, while still allowing their data to be combined as part of a relation. Entities in separate tables are commonly related through a *foreign key*—an attribute of a record indicating where to find its related record. Tables of related entities can be combined through a *join*, which creates a new temporary table where each record corresponds to a combination of records from the original tables, joined together by a specified pattern for matching the records.

Most RDBMSs implement *SQL*, a special-purpose language for manipulating data held in such systems. SQL is declarative, describing conditions on the relation rather than supplying procedural code that computes the outcome. As such, it is up to the RDBMS to produce an optimized plan to execute a query or other operation (Kifer et al., 2005a). The SQL query language is typically supplemented with vendor-specific extensions to the language. One common feature of RDBMSs not covered by SQL is the management of *indexes*—auxiliary data structures for speeding up queries. An index may take the form of a B-tree or hash table that maps a function of an attribute of a record to the location of that record (Kifer et al., 2005b). With an index a query may jump right to the necessary record, rather than scanning the table sequentially until the record is found.

Since the 1970s, the relational model has been the dominant database paradigm, as an RDBMS can provide very strong guarantees about the consistency and durability of data. In recent years, however, there has been renewed interest in non-relational databases, eschewing the guarantees of an RDBMS in favor of performance or flexibility of data representation (Stonebraker, 2010). One non-relational model that I considered for use in EGTAOnline is the *document-oriented* model. A document-oriented database stores data

as a collection of documents, each of which may have arbitrary key-value pairs. Since there are few restrictions on the types used as values, a document-oriented database allows one to hierarchically nest data. In contrast to the fixed schema of the relational model, a document-oriented database also makes it easy to add unstructured, or semi-structured data. Other popular non-relational database models include graph databases, simple key-value stores, and columnar databases, in which data corresponding to a single column are arranged sequentially on disk. As I argue in the next section, game data is a good fit for the relational model, but other database models may be useful for game-theoretic applications.

4.2 Representing Games in a Database

In EGTA, analysis is typically conducted on a (potentially sparse) payoff matrix. The payoff matrix representation suggests a hierarchical structure to data in which a Game is composed of Profiles, which in turn have a mapping from each player's strategy choice to the payoff that player achieves. Accordingly, I originally tried storing the game data for EGTAOnline in a document-oriented database in which data were embedded similar to the structure of the payoff matrix. One problem with this approach, however, is that such a hierarchical structure impinges on the ability to conduct common analysis tasks. In particular, it is rare that a pure-strategy profile is considered in isolation when conducting game-theoretic analysis. Identifying Nash equilibria, for example, requires comparing payoffs that a player can achieve in the target profile to those they can achieve through unilateral deviation of strategy. As mentioned above, the ability for a profile to be reached through a single-player deviation is a relationship that a payoff matrix encodes spatially, by having each dimension correspond to the strategy choice of one player. The reality of storing a payoff matrix with greater than two players on disk or in memory requires encoding this relationship in some other way, and organizing payoff data into profiles may not be the most useful for a one- or two-dimensional storage medium, such as modern RAM or hard drives.

In Chapter 3 I discuss the role-symmetric representation of games, and how such a representation allows compactly storing a game. In addition to reducing the storage requirements of a game, using a compact representation can also reduce the cost of analysis, provided there is no significant overhead to working in the compact representation, as operations that scan the data set have less data to scan. By virtue of generalizing fully symmetric and fully non-symmetric games, using a role-symmetric representation also allows one to fix a schema for games, regardless of degree of symmetry.

To encode role-symmetric game representations, I designed the schema shown in Fig-

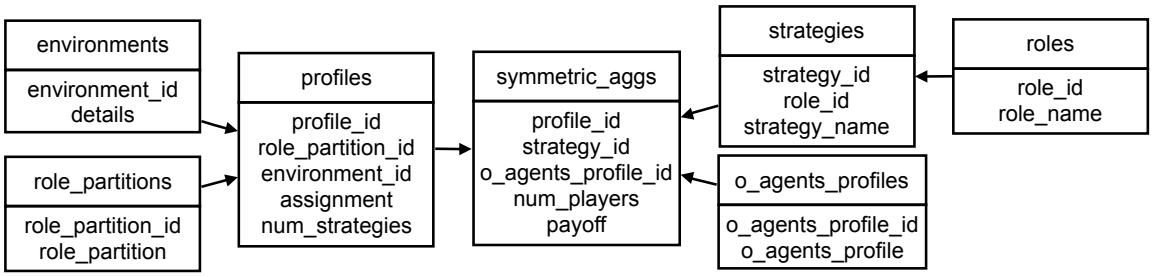


Figure 4.1: Schema for empirical games.

ure 4.1. In this diagram, an arrow indicates a one-to-many relationship: a relationship in which a single record in the source table can be referenced by many records in the target table. These relationships are encoded by foreign keys named according to the pattern "X_id", where "X_id" indicates that the related record can be found in table "X", and will have a matching value for the attribute "X_id".

The primary entities for conducting game-theoretic analysis in the database are the aggregates of players within profiles that share a strategy and payoff, which I term *symmetric aggregations*.[2] A symmetric aggregation is uniquely identified by a profile and a strategy, and has a "num_players" field, which denotes how many players play the specified strategy in the associated profile. Each profile has one symmetric aggregation for each distinct strategy in its strategy assignment. The count of symmetric aggregations associated with each profile is stored in the "num_strategies" field, to simplify several queries. A symmetric aggregation is also associated with an *other-agents profile*—an assignment of strategy that matches the profile, but holds out one player represented by the symmetric aggregation. Symmetric aggregations that share an other-agents profile s_{-i} store the possible payoffs to the held out player i for the deviation set $\mathcal{D}_i(s)$, where s is any profile in $\{(\hat{s}_i, s_{-i}) : \hat{s}_i \in S_i\}$. Identifying the symmetric aggregation with highest payoff among those that share s_{-i} is equivalent to finding i's pure-strategy best response correspondence to s_{-i}, $\mathcal{B}_i(s_{-i})$.

Notably absent from this schema is a table for games. Although the payoff matrix view implies that a profile belongs to a single game, I treat games as views onto the data that enable game-theoretic analysis, as in EGTAOnline. In addition to the benefits espoused in Chapter 3, this approach enables expressing the many-to-many relationship between games and profiles without a junction table to explicitly manage the membership of profiles

[2]In EGTAOnline, the payoff field of a symmetric aggregation is the sample mean for payoffs observed for players playing the associated strategy and profile. In this chapter I do not discuss observations or players as part of the proposed schema as such tables are specific to EGTAOnline's implementation, and are not required for the tasks described in this chapter.

in games. Not storing a junction table is particularly important because games can have huge numbers of profiles, and the membership in a game can change rapidly as the analyst examines different subgames in the course of analysis. As such, using a junction table would require considerable storage and have significant churn.

To implement games as views onto the data, this schema implements several concepts introduced by EGTAOnline. The environments table stores data necessary to determine if the data of two profiles are compatible.[3] For simulation-based EGTA, the "details" field stores the information required to distinguish simulator instances, namely the simulator program and run-time configuration. If instead we are using data from observations of real play, the environment table may store features of the interaction that are needed to classify similar scenarios. For example, in a sponsored-search bidding scenario (Borgers et al., 2013), we may record details about the auction mechanism as well as the associated keyword, since the behavior observed across different keywords may not be comparable.

Another novel concept is that of the *role partition*, which maps each role to the number of players in that role. This is equivalent to a partition of players into roles under the role-symmetric representation, $\{I_r\}$, as the identity of players does not matter when they are symmetric. As I demonstrate in Section 4.3, the combination of environment, role partition, and set of allowed strategies enable querying for subgames of compatible data, the core of game rendering in EGTAOnline.

4.3 Game-Theoretic Primitives

In this section I define SQL queries that correspond to primitive game-theoretic operations. Perhaps the most fundamental operation is assembling subgames required for analysis. To show that a (potentially mixed-strategy) profile is a Nash equilibrium, we need consider only profiles in either the support or the deviation set of the profile. For a pure-strategy profile, the set of profiles in support is simply the profile itself. For a mixed-strategy profile, the set of profiles in support is the subgame with strategies restricted to those played with nonzero probability by at least one player. Listing 4.1 provides an SQL query for profiles matching the subgame specified by the environment_id ":env_id", role_partition_id ":rp_id", and strategies restricted to those with strategy_id in the set ":allowed_strats".[4] The

[3]To simplify exposition, I do not describe what form the data takes for the "details", "assignment", "role_partition", or "o_agents_profile" fields on their respective tables. They may be implemented as human-readable descriptions of their respective entities, as the values of these attributes are not required by any queries that I present in this chapter. The "details" data, however, could be used for clustering profiles based on the similarity of environmental conditions, necessitating a structured representation.

[4]For other SQL listings I omit the environment_id and role_partition_id conditions to simplify presentation. Note also that these conditions can safely be omitted if data gathered for different simulator instances are

```
SELECT profile_id
FROM symmetric_aggs
LEFT JOIN profiles USING (profile_id)
WHERE strategy_id = ANY(:allowed_strats)
AND environment_id = :env_id
AND role_partition_id = :rp_id
GROUP BY profile_id, num_strategies
HAVING COUNT(*) = num_strategies
```

Listing 4.1: Query for subgame

essential idea employed here is that symmetric aggregations encode the strategy assignment of a profile, in that a profile matches the requested subgame only if *all* of its symmetric aggregations match strategies of the subgame. If a profile's assignment includes a strategy that is not part of the desired subgame, then one or more symmetric aggregations will be filtered out by this query since they do not match any of the allowed strategies. The query then counts the number of symmetric aggregations that remain for each profile, and if this count does not match the "num_strategies" field of that profile, the profile is excluded from the result.

Prior to hitting on symmetric aggregations as a key organizing unit, I tried matching a regular expression of the roles and their allowed strategies to string representations of the profiles' assignments. Although this earlier approach worked (in conjunction with a role partition and environment as in Listing 4.1) insofar as it returned the correct set of profiles, it could not be efficiently indexed, and building the correct regular expression was complicated and error prone. In EGTAOnline, I found switching to the method in Listing 4.1 yielded two orders of magnitude improvement over using regular expressions, though no serious attempt was made to optimize either method.

Listing 4.2 gives a SQL query for the deviation set of a pure-strategy profile, specified by ":prof_id". Since an other-agents profile encodes the strategy choices of all but a single player, the deviation set is the union of all profiles that share an other-agents profile with the target profile. This set is obtained by filtering the symmetric aggregations table by the set of o_agents_profile_ids found in the symmetric aggregations of the target profile. The result of this query differs slightly from the description of deviation set given in Chapter 1 in that it includes the profile specified by ":prof_id". Using this approach one may also calculate regret, as in Listing 4.3. When payoff data is missing for profiles in a deviation set, the result of this query is a lower bound on regret, as the missing profile may provide an even greater benefit to deviation.

stored in separate databases.

```
SELECT DISTINCT profile_id
FROM (SELECT o_agents_profile_id
  FROM symmetric_groups
  WHERE profile_id = :prof_id) other_agents
LEFT JOIN symmetric_aggs
USING (o_agents_profile_id)
```

Listing 4.2: Query for the deviation set of a pure-strategy profile

```
SELECT MAX(symmetric_aggs.payoff-other_agents.payoff)
FROM (
  SELECT o_agents_profile_id, payoff
  FROM symmetric_aggs
  WHERE profile_id = :prof_id) other_agents
INNER JOIN symmetric_aggs
USING (o_agents_profile_id)
```

Listing 4.3: Query for regret of a pure-strategy profile

We can similarly query for the deviation set of a mixed-strategy profile by finding the deviation set of each pure-strategy profile in support of the mixed-strategy, as in Listing 4.4. Here, the support is computed using a user-defined function labeled "subgame", which implements Listing 4.1, passing in the set of strategies in support of the mixed strategy, ":strats_in_support", as the set of allowed strategies. As in Listing 4.2, this method returns the union of the support and the deviation set. Calculating regret of a mixed-strategy profile is also possible in SQL, but is significantly more involved. Calculating deviation payoffs requires computing the expected payoff of deviating to each pure strategy, weighting the payoff of each symmetric aggregation that matches that strategy by the probability that its other-agents profile is played by the non-deviating players. Since it is impractical to specify the probability of each other-agents profile being played directly, implementing this calculation could involve computing these probabilities from the probabilities of each strategy being played in the mixed-strategy profile, and storing the results in a temporary table.

4.4 Identifying PSNE

Using the query for regret of a pure-strategy profile (Listing 4.3), one can define an algorithm that identifies all pure-strategy Nash equilibria of a game. This algorithm (here after referred to as IsNash) computes the regret of each profile, and returns those for which

```
SELECT DISTINCT profile_id
FROM (SELECT o_agents_profile_id
  FROM subgame(:strats_in_support)
  JOIN symmetric_aggs
  USING(profile_id)) other_agents
LEFT JOIN symmetric_aggs
USING (o_agents_profile_id)
```

Listing 4.4: Query for the deviation set of a mixed-strategy profile

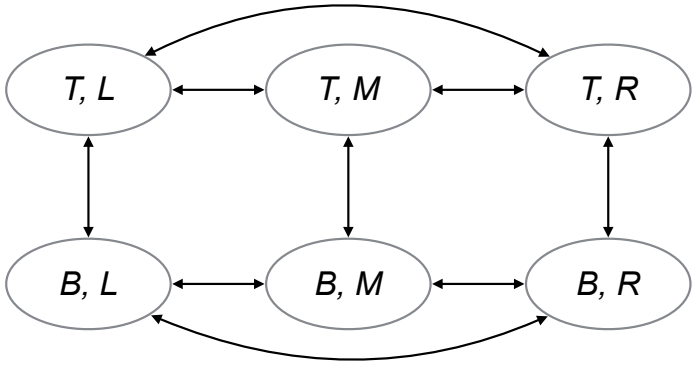


Figure 4.2: Data dependence graph for calculating regret of profiles in an example game.

regret is zero. This is similar to the method of enumerating PSNE found in the most recent version of Gambit (McKelvey et al.); however, IsNash is suboptimal for two reasons. First, it repeats payoff comparisons, failing to exploit the fact that testing whether profile s^A is a beneficial deviation from s^B also tells us whether s^B is a beneficial deviation from s^A. Furthermore, it ignores knowledge that is implied by transitivity, as for a given player and other-agents profile there is a fixed payoff ordering on strategies. Second, this approach leads to non-sequential data access patterns, which can cause poor performance when there is more data than can fit in main memory. As stated previously, the regret of a profile is a function of payoffs for the target profile and all profiles in its deviation set. In other words, computing the regret of a profile depends on payoff data associated with $O(NM)$ other profiles. Figure 4.2 depicts the profile data dependencies for calculating the regret in a two player game where the strategy sets are $\{T, B\}$ and $\{L, M, R\}$. In this graph, an arrow indicates that calculating the regret of the source profile requires payoff information from the target profile. As calculating the regret of each profile in this example depends on data from three other profiles, there is no way to order profiles on disk to enable IsNash to compute the answer by reading the data sequentially, without duplicating the data.

To address these limitations, I present the procedure of Listing 4.5 (hereafter BestResponses). This approach relies on the characterization of Nash equilibrium profiles in terms

```
WITH best_response_payoffs AS (
  SELECT o_agents_profile_id,
    MAX(payoff) AS payoff
  FROM symmetric_aggs
  GROUP BY o_agents_profile_id)
SELECT profile_id
FROM best_response_payoffs
INNER JOIN symmetric_aggs
  USING (o_agents_profile_id, payoff)
INNER JOIN profiles USING (profile_id)
WHERE strategy_id = ANY(:allowed_strats)
GROUP BY profile_id, num_strategies
HAVING COUNT(*) = num_strategies
```

Listing 4.5: Query for pure-strategy Nash equilibria through identifying best responses (BestResponses)

of mutual best response. In a mutual best response, the strategy that the profile assigns to each player maximizes that player's payoff, conditional on the other players playing the strategy assigned to them by the profile. We can thus find all PSNE by finding the symmetric aggregations with a strategy and other-agents profile such that the symmetric aggregation's strategy is a best response to its other-agents profile, and label as equilibria the profiles for which all of their symmetric aggregations meet this condition.

The common table expression, "best_response_payoffs", returns a mapping from other-agents profiles to the maximum payoff that the held-out player can achieve. Matching symmetric aggregations against this expression returns the set of strategies that are best responses to each other-agents profile, allowing for ties. Allowing ties means that the returned set may include profiles where agents are indifferent between playing their strategy in the profile and another of their strategies. Such profiles are known as *weak Nash equilibria*. To return the set of *strict Nash equilibria*—profiles in which payoff is strictly decreased through deviation—we can filter out entries in "best_response_payoffs" that have more than one matching symmetric aggregation.

By organizing comparisons around identifying best responses, duplication of comparisons is eliminated; the set of payoffs corresponding to an other-agents profile is searched only once to identify the best-response to that other-agents profile. Whereas IsNash performs $O(NM)$ work for each of M^N profiles, BestResponses performs $O(M)$ work for each of NM^{N-1} other-agents profiles, reducing worst case complexity by a factor of M for non-symmetric games.[5] BestResponses also allows data to be organized such that it can

[5]For arbitrary role symmetry, the number of payoff comparisons conducted by IsNash

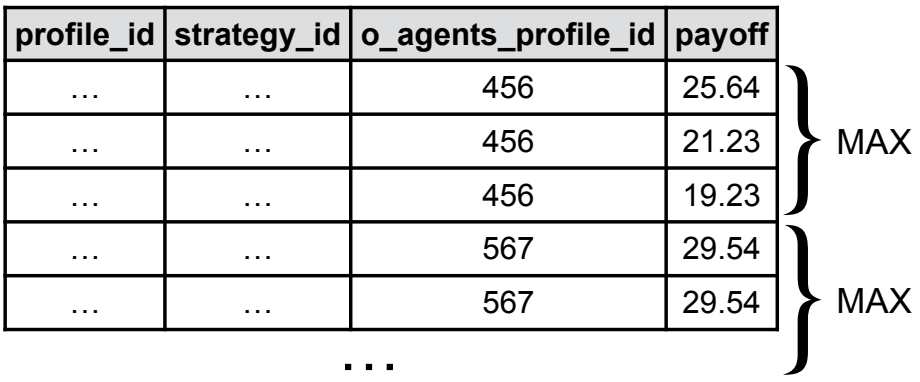


Figure 4.3: Finding best responses in the symmetric aggregations table.

be accessed sequentially, performing all payoff comparisons in a single pass, as demonstrated in Figure 4.3. For this diagram, imagine the data on disk, one record after another. Since symmetric aggregations can be grouped together by o_agents_profile_id, in one sequential pass over the data, the symmetric aggregations that are best responses to their o_agents_profile_id can be identified, and their profile_ids recorded. Not all relational database management systems explicitly support maintaining an order to the data on disk; in such cases the data may still be processed in a single pass by using a hash table to record the highest payoff seen thus far for each other-agents profile. In contrast, determining if each profile is a Nash equilibrium in profile order requires gathering payoff data from various parts of the table, regardless of how the payoff data is ordered. I/O performance may be improved over the profile-oriented approach, particularly when games are too large to fit in main memory.

4.5 PSNE-Finding Performance

4.5.1 Comparison of SQL Algorithms

Figure 4.4 presents a run-time comparison of IsNash and BestResponses implemented in PostgreSQL 9.3.2, a popular open source RDBMS. For IsNash, processing each profile references only a small portion of the total data, and thus indexes were added to speed up finding symmetric aggregations that correspond to the deviation set of a profile. Since

is $\left(\sum_{r\in R} M_r\right)\left(\prod_{r\in R}\binom{N_r+M_r-1}{N_r}\right)$ and the number of comparisons for BestResponses is $\sum_{r\in R}\left(M_r\binom{N_r+M_r-2}{N_r-1}\prod_{r'\in R, r'\neq r}\binom{N_{r'}+M_{r'}-1}{N_{r'}}\right)$. For a fully symmetric game with N players and M strategies, these expressions simplify to $M\binom{N+M-1}{N}$ and $M\binom{N+M-2}{N-1}$ respectively, meaning that IsNash performs more comparisons in this setting by a factor of $1+\frac{M-1}{N}$.

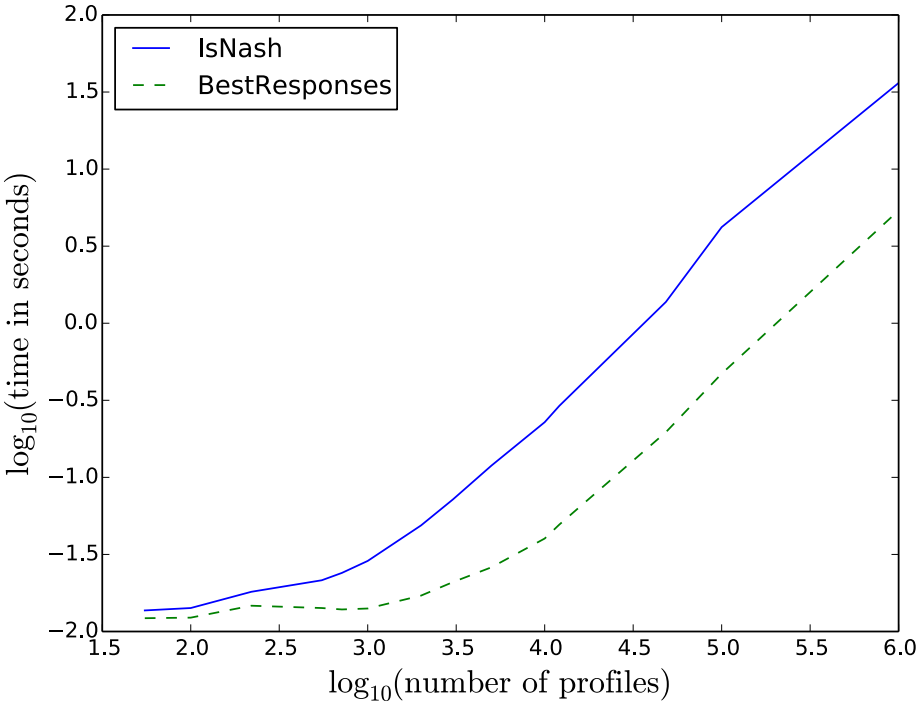


Figure 4.4: Performance of two SQL methods for identifying all PSNE.

BestResponses performs a constant number of full table scans, rather than random accesses, I did not find any indexes to be beneficial.[6]

For this experiment[7] I examined uniformly random games with 2–6 players and three different levels of symmetry—fully symmetric, fully non-symmetric, and splitting the players into two (approximately) equal roles—with each role having 10 strategies to play. This produced a spectrum of game sizes, from 55 to a million profiles. Each point in the graph corresponds to the mean performance over 30 game instances, with both algorithms operating on the same data. From this graph we observe that BestResponses outperforms IsNash over a spectrum of game sizes and degrees of symmetry.

4.5.2 Comparison to Gambit

To assess the benefits of conducting analysis in the database, I compare the performance of BestResponses to Gambit. Gambit's PSNE enumeration algorithm is given by Algorithm 4.1 (hereafter GambitPSNE). In contrast to IsNash, GambitPSNE avoids some comparisons by progressing to the next profile as soon as a beneficial deviation from the pre-

[6]When more than one game is stored in the database, as in EGTAOnline, an index for finding symmetric aggregations that match a set of profiles could improve query performance.

[7]This experiment was conducted on a 2.7 GHz Intel quad-core processor with 12 GBs of RAM and a 7200 RPM hard drive, using a mildly optimized PostgreSQL configuration.

vious profile is found. As discussed above, this algorithm may be improved upon by computing best responses to other-agents profiles, as in Algorithm 4.2 (hereafter GambitBR). GambitBR differs from BestResponses in that it, like GambitPSNE, may avoid some payoff comparisons. In the case of GambitBR, as it iterates through players it removes a profile from its candidate set if the profile is not a best response for the current player, potentially leading to fewer payoff comparisons in each step. To assess the performance of the two algorithms, and compare both against BestResponses, I implemented GambitBR within the Gambit codebase.

Algorithm 4.1 Gambit PSNE Enumeration (GambitPSNE)

Require: Γ, the game to search
  $E \leftarrow \{\}$
  for $s \in \Gamma$ **do**
    if IsPSNE(s) **then**
      $E \leftarrow E \cup \{s\}$
    end if
  end for
  return E

  procedure IsPSNE(s)
    for $i \in I$ **do**
      for $\hat{s}_i \in S_i$ **do**
        if $u_i(\hat{s}_i, s_{-i}) > u_i(s)$ **then**
          return false
        end if
      end for
    end for
    return true
  end procedure

I compared the performance of the three algorithms under two scenarios: a "warm cache" scenario intended to capture performance under optimal conditions, and a "cold cache" scenario capturing less than optimal, but still common conditions. For the warm cache scenario, all algorithms were run once prior to the measured run, so as to have useful data already resident in main memory. For the database implementation, this is obviously beneficial, as most modern databases hold recently accessed data in main memory to speed up subsequent queries on the same data. When the size of the database does not exceed the available memory, it is not uncommon for all of the available data to be present in RAM at the time of querying. Gambit on the other hand has no equivalent capabilities, though it is possible to conduct many different analyses on the same data by using Gambit as a library. For the warm cache scenario, I measured only the time spent inside the algorithm

Algorithm 4.2 Gambit Mutual Best Response Enumeration (GambitBR)

Require: Γ, the game to search

```
E ← {}
j ← an element of I          ▷ First find the set of profiles where j plays a best response.
for s_{-j} ∈ Γ do
    for s_j ∈ B_j(s_{-j}) do
        E ← E ∪ {(s_j, s_{-j})}
    end for
end for
for i ∈ I, i ≠ j do
    for s ∈ E do
        if not IsBestResponse(i, s) then
            E ← E \ {s}
        end if
    end for
end for
return E

procedure IsBestResponse(i, s)
    for ŝ_i ∈ S_i do
        if u_i(ŝ_i, s_{-i}) > u_i(s) then
            return false
        end if
    end for
    return true
end procedure
```

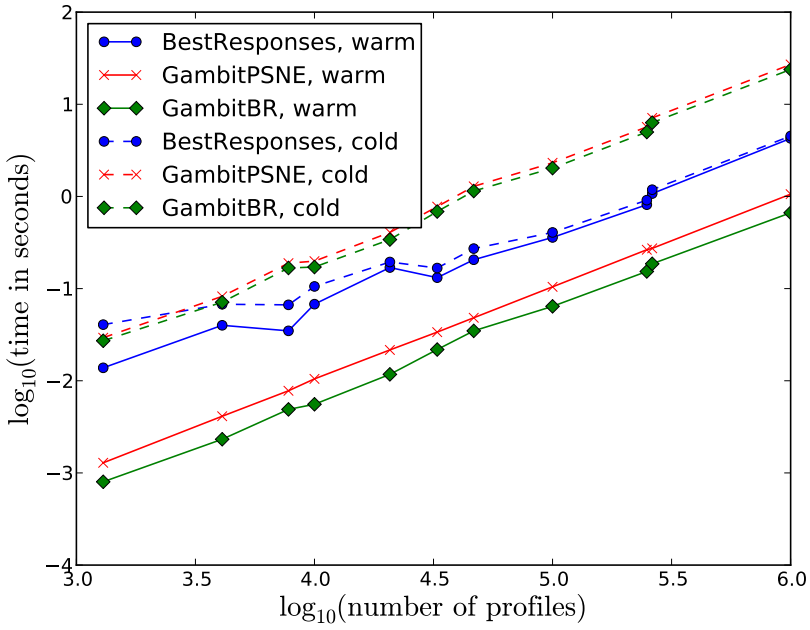


Figure 4.5: Comparing performance of PSNE finding methods under two cache scenarios.

for GambitPSNE and GambitBR, and not any time spent building the game representation. For the cold cache scenario, before each run I dropped all system caches to ensure that no useful data was already present in memory prior to measuring performance. For this scenario, I measured the total time spent by GambitPSNE and GambitBR when invoked from the command line. This mirrors one of the most typical use cases for Gambit, in which a particular analytical method is invoked through a command line interface, passing in the location of a file representation of a game.

For each scenario, I tested the performance of these algorithms on fully non-symmetric games with 4–6 players, with 6, 8, 10, or 12 strategies for the 4- and 5- player games, and 6, 8, or 10 strategies for the 6-player games.[8] This produces a spectrum of game sizes between 1296 and one million profiles. Although BestResponses can operate on games with arbitrary degrees of role symmetry, Gambit does not exploit role symmetry, and thus I limit my test set to non-symmetric games. As in the previous experiment, for each game setting, 30 uniformly random game instances are constructed, all algorithms operate on the same set of data, and I use mean performance as my metric. The results of these experiments are presented in Figure 4.5.

For the warm cache setting (graphed in solid lines), both Gambit variants outperform

[8]These experiments were conducted on a 3.5 GHz Intel quad core CPU, with 32 GB of RAM and a solid-state drive, using a mildly optimized PostgreSQL configuration.

the database algorithm, BestResponses. This is not surprising as Gambit is implemented in a performant language (C++), and does not have the overhead of providing all the functionality of a database. Of minor note is that GambitBR, my implementation of the mutual best response algorithm in Gambit, outperforms the stock Gambit algorithm, GambitPSNE. Although this may have been predicted from examining the worst-case complexity of each algorithm, given that the number of comparisons required by both algorithms is heavily test-case dependent, this outcome was not guaranteed by algorithmic analysis. As a consequence of their ability to avoid unnecessary comparisons, neither Gambit algorithm demonstrated as strong a correlation with the number of strategies as BestResponses: the two big dips in the curve for BestResponses occur when transitioning to a larger number of players and smaller number of strategies.

In the cold cache setting (graphed in dotted lines), the database algorithm, BestResponses, outperformed the Gambit algorithms for all but the smallest games considered. By comparing the two settings, we can see that this reversal in outcomes is due to a difference in costs for making data usable by the algorithms. While the database approach had a shrinking overhead to loading data relative to the size of the game, Gambit's time spent building an in-memory representation exhibits approximately the same order growth as the time spent in the algorithm. Furthermore, this representation-building overhead in Gambit is the dominant cost for this analysis task: for the 6-player, 10-strategy games, GambitPSNE spends just over a second finding all PSNEs of the game, but spends more than 25 seconds building its in-memory representation. For more expensive analysis, such as computing mixed-strategy Nash equilibria, this overhead may be less significant. For very data-intensive tasks such as control variate adjustment or bootstrap computation, however, it is worth bearing such costs in mind.

4.6 Incremental Analysis

By using a database as a storage mechanism, it is possible to automatically conduct simple analysis as data becomes available and store the results alongside the data. This capability enables two valuable outcomes: the interim analysis may provide useful feedback to the analyst for use in sample control, and the cost of conducting analysis may be partially amortized over the time spent gathering data. In this section I discuss two tasks which benefit from this capability: computing the regret of profiles, and tracking the set of maximal complete subgames.

4.6.1 Incremental Regret Maintenance

Rather than finding the set of pure-strategy Nash equilibria as a query that touches all of the data, we can add a "regret" field to profiles that we update as observations update our payoff estimates. To maintain up-to-date regret values, on any change to the payoffs of profile s, we must update the regret value for profile s, and check for required changes to the regret values of each profile in the deviation set of s. Although this could require updating many more rows on each observation, it would enable roughly constant-time querying for the set of pure-strategy Nash equilibria by indexing on regret. Additionally, incrementally maintaining regret in this way would enable a user to query for the set of pure-strategy profiles that are δ-Nash equilibria, for a specified δ, in logarithmic time.

As discussed in the previous chapter, when the analyst has control over what data is to be gathered, it is often advantageous to condition the selection of profiles to observe on knowledge of the observations taken thus far. One algorithm discussed in Section 3.6.1, Minimum-Regret-First Search (MRFS) (Jordan et al., 2008), is particularly apt for sequentially updated regret estimates. MRFS attempts to increase the regret lower bound of the profile that currently has the minimal lower bound on regret in each step by evaluating a profile in its deviation set that has not yet been observed. As mentioned in Section 4.3, the lower bound on regret is precisely the value returned by my proposed regret query when the database is missing payoff information for profiles in the deviation set. Sequential sampling algorithms that target statistical measures—the topic of the next chapter—as well as profile exploration algorithms that rely on other payoff statistics, may be similarly supported by storing the appropriate statistics and triggering their recalculation on the arrival of data.

4.6.2 Identifying Maximal Complete Subgames

In order to compute an mixed-strategy Nash equilibrium (MSNE) of a game (or subgame), one must be able to estimate the utility to all players for every pure-strategy profile. In the EGTA methodology, this generally means that an MSNE can only be computed on *complete subgames*: subgames for which all profiles have been observed. A common preprocessing step for computing MSNEs on empirical game data sets is to identify the set of subgames that meet this condition, and are *maximal* in the sense that adding another strategy would include a profile in the subgame that has not been observed. Such subgames are sometimes also referred to as *maximal cliques* (Wellman et al., 2005).

Identifying the set of maximal complete subgames for a game is computationally expensive, as the set of possible subgames is given by the power set of the set of strategies, and every profile must be checked to see if it has been observed. As replicator dynamics,

a common search method for finding symmetric Nash equilibria, can often converge quite quickly, I have observed that the major cost of analysis can be identifying the subgames to search, rather than the search itself. Fortunately, the set of maximal complete subgames can be maintained and updated incrementally, as each time a previously unobserved profile is observed, we can update our knowledge of which subgames are complete. To address this task, I have specified the Incremental Maximal Complete Subgame Update (IMCSU), presented in Algorithm 4.3.

Algorithm 4.3 Incremental Maximal Complete Subgame Update

Require: s, the previously unobserved profile
Require: τ, the subgames database table
  $\zeta \leftarrow$ the ordered set of distinct strategies in s
  if a subgame with strategies $= \zeta$ is not present in τ **then**
    Insert $(\zeta, |\text{SUBGAME}(\zeta)|, \text{REQUIRED\_COUNT}(\zeta))$ into τ
  end if
  for q in τ such that $q_{strategies} \supset \zeta$ **do**
    $q_{observed} \leftarrow q_{observed} + 1$
    if $q_{observed} = q_{required}$ **then**
      for p in τ such that $p_{strategies} \subset q_{strategies}$ **do**
        Delete p from τ
      end for
    end if
  end for

IMCSU maintains information about subgame completeness in a table τ. Each entry in τ corresponds to a subgame and has the following fields:

strategies The set of strategies associated with the subgame

observed The number of profiles in the subgame that have been observed

required The number of profiles required for the subgame to be complete

IMCSU is to be invoked whenever the first observation is taken of a profile, and creates a entry in τ matching the strategy set of the profile if one does not yet exist. The "observed" field is set using the subgame query from Listing 4.1 to count the number of existing profiles that form part of the subgame, and "required" is set to $\text{REQUIRED\_COUNT}(\zeta)$, the number of profiles in the profile space defined by the strategy set ζ.[9] Next, every subgame in the

[9]To simplify presentation, Algorithm 4.3 is presented under the assumption that all the data in the database is compatible, as defined in the previous chapter. In general, the subgame table would also require "environment_id" and "role_partition_id" fields to track data incompatibilities; both pieces of information would also be necessary for computing the number of profiles that match a subgame, and the REQUIRED_COUNT function would require a role partition as an additional argument.

table that the profile matches (those which have strategy sets that are strict supersets of ζ) has its "observed" field incremented. If "observed" and "required" are equal for an entry in τ, we know that it is complete subgame. To enforce that the only complete subgames in the table are also maximal, once a subgame q becomes complete, all entries in the table with strategy sets that are strict subsets of $q_{strategies}$ are removed. Since the size of the profile and symmetric aggregations tables must be greater than the subgame table (as a subgame record cannot be constructed without first observing a new profile, and not all profiles require a new subgame record to be constructed), the most expensive operation in this algorithm is querying for the number of profiles the match the subgame, which has cost linear in the size of the (admittedly large) symmetric aggregations table.

Querying τ for subgames that have their "observed" value equal to their "required" value returns the set of maximal complete subgames. With proper indexing, this set can be returned in roughly constant time. Querying for the set of maximal complete subgames with strategies from some restricted set, however, is less straightforward. Such an operation may be desired if the analyst wishes to find MSNEs in a specially restricted subgame, as in the case of the market maker example in Section 3.5. To gather the set of maximal complete subgames in this situation, the Maximal Complete Subgames with Restricted Strategies (MCSRS) algorithm, presented in Algorithm 4.4, may be employed.

Algorithm 4.4 Maximal Complete Subgames with Restricted Strategies

Require: S', the restricted set of strategies
Require: τ, the subgames database table
  $Z \leftarrow \mathcal{P}(S')$
  $Q \leftarrow \{\}$
  while $Z \neq \{\}$ **do**
    $\zeta \leftarrow$ the largest element of Z
    if there exists q in τ such that $q_{strategies} \supseteq \zeta$ and $q_{observed} = q_{required}$ **then**
      $Z \leftarrow Z \setminus \mathcal{P}(\zeta)$
      $Q \leftarrow Q \cup \{\zeta\}$
    else
      $Z \leftarrow Z \setminus \{\zeta\}$
    end if
  end while
  return Q

MCSRS iterates through the set of possible subgames (given by the power set of the restricted strategy set S'), in order from largest to smallest, querying the database for maximal complete subgames that subsume the current subgame. Due to the maximality condition, when such a subgame is identified, all subgames that it subsumes are removed from consideration, and the current subgame is added to Q, the set of maximal complete subgames

for the restricted strategy space. Since MCSRS must iterate through the power set of the restricted strategy set, the cost of MCSRS grows exponentially in the size of the restricted strategy set. Though exponential, this approach still benefits from the cost amortization of IMCSU as the cost of MCSRS does not depend on the size of the game data set.

4.7 Discussion

This chapter constitutes a first attempt at mapping game-theoretic analysis to the database. By moving analysis closer to the data we may reduce the cost spent in translating between different representations. As I demonstrated in this chapter, for Gambit, a common game-theoretic analysis software package, this translation cost can be quite significant. A further benefit of this approach is that it supports using analysis as feedback for the sampling process, rather than simply an end product. Additionally, since gathering data often occurs over a long period of time, having analysis in the database can enrich the data automatically, amortizing the cost of some analysis. In this vein, I proposed an algorithm for maintaining information about maximal complete subgames as observations are taken, reducing the cost of this operation at analysis time.

I designed a novel schema supporting game-theoretic analysis. The constraints of working in the database drove me toward new entities as the guiding organization of the data: *symmetric aggregations* and *other-agents profiles*. Though these entities are always implicitly present in game descriptions, in the database they come to the fore, as they enable quickly determining a profile's membership in a game, a support, or a deviation set. With my schema I was able to develop SQL implementations for important game-theoretic operations, such as calculating regret and identifying PSNEs. When implemented in a common RDBMS, my PSNE-identifying algorithm was sufficiently performant that I observed significant time saving relative to using the Gambit command line interface for the same task. Furthermore, this algorithm is generic with respect to role symmetry, providing further performance benefits for games with symmetry over conversion to the non-symmetric representation of Gambit.

Of great interest for future work is the ability to find mixed-strategy Nash equilibria of large games. Existing algorithms for this task do not map well to SQL, as they typically require iteration or solving mathematical programs; however, a hybrid approach, in which the database is used to efficiently query for the data needed in each step of analysis, as in the MADlib Analytics Library (Hellerstein et al., 2012), could be beneficial. Furthermore, other data technologies, such as MapReduce, may be employed to take advantage of opportunities for parallelism in these algorithms (Widger and Grosu, 2008). Although

few algorithms are designed to accommodate role-symmetric representation of games, the required modifications may be limited as methods exist for games that are fully symmetric (Cheng et al., 2004).

It would also be interesting to explore alternative database paradigms, and compare performance to the relational model. As mentioned, I had previously attempted to model games in a document-oriented database with little success, but a columnar database may further improve I/O performance for analytic tasks. A graph database could also yield benefits, as the "reachable through single player deviation" relationship can be encoded as a graph. For some games, such as those that can be more compactly encoded as an action-graph game (Jiang et al., 2011), there are specialized algorithms that can solve the game faster by operating on its graph.

CHAPTER 5

Bootstrap Methods for Sequential Estimation of Nash Equilibria

Since EGTA builds game models by estimating utility functions from noisy observations, solutions to these models carry uncertainty. In many EGTA studies, however, approximate solutions are presented without any quantification of statistical confidence. Furthermore, the exact method used for sampling profiles is rarely specified, and raw observation data is rarely provided. Even if such information were provided, there are no agreed-upon methods for evaluating the statistical evidence acquired through EGTA. When publishing equilibrium analysis, we would like to provide readers with an assessment of the likelihood that the equilibria presented are close approximations of Nash equilibrium when played in the true game.

Statistical quantification of EGTA claims may have been lacking in the past due to difficulty in accurately evaluating the complex hypotheses that arise in this methodology. In particular, classical parametric statistical methods are ill-suited for evaluating the primary statistic of interest, regret. In order to use a parametric method, the distribution family for the statistic must be known, and most methods are only designed to work on well-known distributions such as the Gaussian distribution. The sampling distribution of regret—the distribution of regret measured in an empirical game constructed from a finite sample—however, has no closed form, as regret in an empirical game is typically computed by taking the maximum of sample means from independent, non-identical distributions. Many statistical methods also rely on the size of the sample being fixed, as they were developed during a time when a statistician was typically not the same person that gathered the data (Ghosh and Sen, 1991). As such, EGTA's iterative approach to model construction further complicates statistical evaluation, as the number of observations allocated to each profile is generally not fixed, and the data set may be evaluated many times before sampling is terminated.

Difficulties in evaluating the available statistical evidence may lead researchers to con-

clude sampling when equilibrium candidates pass an "eye test"—the candidates do not appear to have beneficial deviations when comparing payoff sample means, and the payoff sample variances are not so high as to cause concern. Alternatively, one may continue sampling until the set of equilibrium candidates, and the probability distributions therein, undergoes minimal change with the addition of further observations. These ad hoc stopping rules have two significant flaws. First, they do not allow us to express statistical confidence that the points identified are (δ-)Nash equilibria of the true game, and this may cause concern about the repeatability of results from simulation. Secondly, these stopping rules can lead to insufficient or excess sampling. Since statistical confidence is never calculated under these stopping rules, they may report an equilibrium candidate for which there is insufficient evidence to claim that the candidate is a (δ-)Nash equilibrium of the true game, or may continue sampling long after accruing sufficient evidence to establish such a claim statistically. When a stopping rule is overly conservative, time and computation are spent gathering observations that do not significantly change the conclusions of the study; such resources could more profitably be allocated toward exploring more strategies or conducting additional experiments.

A further potential concern is that, even if we could evaluate the outcome of these stopping rules statistically, we could be biased in favor of identifying selected profiles as (δ-)Nash equilibria due to sequential sampling (Whitehead, 1986). The concern here is that, when sampling until some criteria are met, the decision to stop is influenced by the order in which one observes the data. Consider for example two sequences of observations, $A = (0, 0, 0, 0, 0)$ and $B = (1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0)$, observed by a stopping rule that samples sequentially until it is either confident that the mean of the generating process is less than 0.5 or when it has made 12 observations. In both sequences, five 0s are observed consecutively, but for sequence A, this run of 0s leads the stopping rule to conclude sampling early, accepting a hypothesis that sequence B, with the benefit of more data, does not support. Not all sequential sampling procedures introduce such bias, but it is important to verify the accuracy of statistical analysis conducted on observations gathered with a sequential stopping rule.

Recently my colleagues developed a method for estimating statistical confidence in regret for empirical games by using a non-parametric statistical method known as the *bootstrap*.[1] Using this method, I construct algorithms for two tasks:

1. sequentially sampling to determine if a given equilibrium candidate is a δ-Nash equilibrium at a specified level of statistical confidence, and

[1]The introduction of the bootstrap for use with empirical game statistics, as well as much of the material from this chapter, was published at AAMAS 2014 (Wiedenbeck et al., 2014).

2. sequentially sampling until a profile is identified that is a δ-Nash equilibrium at a specified confidence level.

The first task can be thought of as a labeling task, where the algorithm samples until it is confident in the appropriate label (as δ-Nash equilibrium or not) for the equilibrium candidate. The second task is a search task, where observations are taken until the algorithm can find a profile for which we have sufficient evidence to declare it a δ-Nash equilibrium with high confidence. To address the potential for bias described above, I experimentally validate that these algorithms are accurate with respect to their stated level of confidence on several synthetic and simulation-based game data sets. I similarly test the accuracy of using the bootstrap to construct confidence intervals when using two common rules of thumb; these experiments help establish whether it is valid to use the bootstrap to estimate confidence in the conclusions of earlier studies that employed ad hoc sampling approaches. Finally, I experimentally compare the performance of the bootstrap-based stopping rule to the two rules of thumb at the task of finding δ-Nash equilibria of the true game.

5.1 The Bootstrap

The bootstrap is a computational method for estimating the sampling distribution of summary statistics (Davison and Hinkley, 1997). Sampling distribution refers to a distribution of a statistic when computed on a finite sample of a given size. Unlike many classical statistical methods, the bootstrap is non-parametric, not relying on any assumptions about the shape of the underlying distribution. As mentioned above, this feature is critical for game statistics as the primary statistic of interest, regret, does not match a known distribution.

The bootstrap is based upon treating a sample Θ as representative of the true population. From this sample, *resamples* are constructed by drawing with replacement $|\Theta|$ times from Θ, simulating the process of taking many independent samples from the true population. For each resample $\hat{\Theta}$, we can compute the statistic of interest, producing an empirical estimate of the sampling distribution of the statistic by repeating this process many times. From quantiles of this estimated sampling distribution we can construct confidence intervals for the statistic.

In applying the bootstrap to empirical game statistics, one must choose how to define a sample Θ and its elements, $\theta \in \Theta$. In constructing a game, observations of different profiles typically belong to different, independent samples, yet are analyzed in concert in EGTA. The bootstrap procedure for EGTA estimates the sampling distribution of a game statistic through computing the statistic on a large number of *bootstrap games* which may be constructed in any of the following ways:

I For each profile s and player i, define Θ_{s_i} to be the set of payoff observations for player i in profile s. For each Θ_{s_i}, draw with replacement from Θ_{s_i} to construct $\hat{\Theta}_{s_i}$. Construct a bootstrap game by setting the payoff to player i for profile s by the sample mean of $\hat{\Theta}_{s_i}$ for all i and s.

II For each profile s, define Θ_s to be set of payoff vectors observed for profile s. For each Θ_s, draw with replacement from Θ_s to construct $\hat{\Theta}_s$. Construct a bootstrap game by setting the payoff to player i for profile s by the sample mean of the payoff to i in $\hat{\Theta}_s$ for all i and s.

III For each profile s, index the observed payoff vectors sequentially. For each index j, define a sample game Γ_j to be a payoff matrix with payoff vector for each profile given by the payoff vector of that profile having index j. Define $\Theta = \{\Gamma_j\}$ to be the set of sample games. Draw with replacement from Θ to create a new set of sample games $\hat{\Theta}$. Construct a bootstrap game for $\hat{\Theta}$ by setting the payoff to player i in profile s to the sample mean of the payoff to i in s across the samples games in $\hat{\Theta}$ for all i and s.

Drawing with scheme II is most similar to the way game data is generated by a simulator, as it preserves correlation among payoffs within a single observation of a profile. Such correlation exists as all players in a profile interact in the same simulation environment, and thus are all affected by the realization of random variables that govern the simulation. This contrasts with scheme I, where payoffs within the same profile are drawn independently, and scheme III, which induces otherwise non-existent correlation between the observations of different profiles, as all observations sharing an index are drawn together during resampling. The number of redraws under scheme II grows with the number of profiles, as a resample must be conducted for every Θ_s. Scheme III, in contrast, performs the same number of redraws regardless of the number of profiles in the game, as the only set that is resampled is the set of sample games. Scheme III does, however, require having the same number of observations for every profile.

In as yet unpublished experiments, my colleagues found that the correlations induced under scheme III do not impinge on the accuracy of the methods; in the name of expediency, I use scheme III throughout, except where explicitly mentioned. The explicit procedure for calculating confidence intervals for regret that I use is as follows:

1. Given a set of observations for each profile, create the set of sample games $\Theta = \{\Gamma_j\}$.
2. Redraw from Θ to produce $\hat{\Theta}$ as in scheme III.
3. Create a bootstrap game $\mathcal{M}$ from $\hat{\Theta}$ by computing the payoff vectors for each profile through averaging their observations.

4. Calculate the regret of the profiles of interest under the model $\mathcal{M}$.

5. Repeat steps 2–4 many times to compute sampling distribution estimates of regret for the profiles of interest.

6. From these distributions, one-sided confidence intervals of level $\gamma \in [0, 1]$ can be constructed by taking the 100γ percentile of the distribution as the right side, setting the left side to zero. Two-sided confidence intervals are similarly constructed for the level γ by taking the $100(\frac{1-\gamma}{2})$ and $100(\frac{1+\gamma}{2})$ percentiles.

Wiedenbeck et al. (2014) experimentally demonstrated that this procedure provides accurate confidence intervals for regret on a variety of synthetic and simulated game datasets. In the subsequent text, ONE-SIDED-REGRET-CI$(\sigma, \Theta_{seq}, \gamma)$ and TWO-SIDED-REGRET-CI$(\sigma, \Theta_{seq}, \gamma)$ compute the one-sided and two-sided bootstrap confidence intervals of level γ for regret of σ, using the observation data Θ_{seq}.

5.2 Using the Bootstrap in Sample Control

By computing bootstrap confidence intervals on regret for candidate equilibria we can provide a measure of the statistical likelihood that the profiles closely approximate Nash equilibrium. For a fixed number of observations, we cannot be guaranteed to have any profiles for which we have statistical confidence that their regret is below a pre-specified threshold δ. In particular, when a confidence interval for a candidate includes the threshold δ, we are unable to distinguish whether the candidate is or is not a δ-Nash equilibrium. Rather than report that we are uncertain whether or not a profile is a δ-equilibrium, we would often prefer to continue sampling until we have sufficient confidence to make a determination. To address this issue, I developed the Confidence-Interval-Based Stopping Rule (CIBSR) presented in Algorithm 5.1.

CIBSR is similar to the repeated confidence interval approach to terminating clinical trials proposed by Jennison and Turnbull (1984), but utilizes the bootstrap to construct confidence intervals in place of parametric assumptions. CIBSR takes as arguments a candidate profile σ, and any observations taken thus far Θ_{init}, and samples sequentially until there is sufficient evidence to decide whether or not the candidate profile is a δ-equilibrium of the true game. CIBSR is parameterized by the acceptable regret threshold δ, the confidence interval level γ, and the number of observations x to gather of each relevant profile in each step. CIBSR decides if a candidate is an δ-equilibrium by comparing the boundaries of a two-sided confidence interval to δ, accepting the hypothesis that σ is an δ-equilibrium

Algorithm 5.1 Confidence-Interval-Based Stopping Rule

Require: σ, the profile to evaluate
Require: Θ_{init}, the observations used to identify σ as an equilibrium candidate
Require: δ, the acceptable approximation threshold
Require: γ, the desired confidence level
Require: x, the number of observations to take of each profile in each step
  $\Theta_{seq} \leftarrow \Theta_{init}$
  $[\delta_{left}, \delta_{right}] \leftarrow$ TWO-SIDED-REGRET-CI$(\sigma, \Theta_{seq}, \gamma)$
  while $\delta_{left} < \delta$ and $\delta_{right} > \delta$ **do**
    Gather and append x new observations of each $s \in \mathcal{S}(\sigma) \cup \left(\bigcup_{\hat{\sigma} \in \mathcal{D}(\sigma)} \mathcal{S}(\hat{\sigma})\right)$ to Θ_{seq}
    $[\delta_{left}, \delta_{right}] \leftarrow$ TWO-SIDED-REGRET-CI$(\sigma, \Theta_{seq}, \gamma)$
  end while
  return $\delta_{right} \leq \delta$

when the interval falls entirely within $[0, \delta]$, rejecting it when the interval falls entirely within (δ, ∞), and otherwise requesting further observations. Sampling under CIBSR is restricted to profiles that can affect the estimated regret distribution of the candidate. These profiles belong to the support of either σ, $\mathcal{S}(\sigma)$, or the deviation set of σ, $\bigcup_{\hat{\sigma} \in \mathcal{D}(\sigma)} \mathcal{S}(\hat{\sigma})$.

Practitioners may begin with no specific candidates, but wish to sample sequentially from a simulator with the purpose of finding one or more equilibria of the true game. For this use-case, I incorporate bootstrap confidence intervals into a sequential equilibrium finding procedure, Confidence-Interval-Based Equilibrium Finding (CIBEF), presented in Algorithm 5.2.

Algorithm 5.2 Confidence-Interval-Based Equilibrium Finding

Require: δ, the acceptable approximation threshold
Require: γ, the desired confidence level
Require: x, the number of observations to take of each profile in each step
  $\Theta_{seq} \leftarrow \{\}$
  $E \leftarrow \{\}$
  while $E = \{\}$ **do**
    Gather and append x new observations of each profile to Θ_{seq}
    for $\sigma \in$ EQUILIBRIA(Θ_{seq}) **do**
      if ONE-SIDED-REGRET-CI$(\sigma, \Theta_{seq}, \gamma) \leq \delta$ **then**
        Append σ to E
      end if
    end for
  end while
  return E

At each step, CIBEF requests x additional observations of each profile and finds equilib-

ria of the updated empirical game. For all experiments in this chapter, EQUILIBRIA(Θ_{seq}) uses replicator dynamics to find symmetric mixed-strategy Nash equilibria of a game constructed from Θ_{seq}. For each equilibrium of the empirical game, a one-sided regret confidence interval at the γ-level is constructed, and if the right-hand side of this interval is not greater than δ, the profile is appended to the set of equilibria. When one or more equilibria of the empirical game meet this criterion, sampling is terminated and the set of candidates meeting the criterion is returned.

5.3 Experimental Data Sets

To evaluate these algorithms, I use two synthetic data sets—randomly generated uniform symmetric games (uSym) and congestion games (Cgst)—and one data set from an existing simulation-based game, the credit network (CredNet) formation scenario described by Dandekar et al. (2012). For each trial from a synthetic data set I first generate a random uniform symmetric or congestion game to act as the true game. When a sampling rule requests observations from a game, these observations are generated by taking the payoffs from the true game and adding zero-mean Gaussian noise to them, with noise drawn independently for each player. To generate a true game from the uSym class, I draw a value from the distribution $U[0, 100]$ for each unique payoff in a symmetric game with 4 players and 4 strategies. To generate a true game from the Cgst class, I use 5 players and 3 strategies; each strategy corresponds to selecting one of three resources with a base value $v_b(s) \sim U[0, 3]$, a congestion cost that is linear in the number of players using the resource $v_l(s) \sim U[0, 1]$, and a congestion cost that is quadratic in the number of players, $v_q(s) \sim [0, 1]$. The payoff to a player from using resource s is a function of the total number $n(s)$ of players choosing that resource: $u(s) = v_b(s) - v_l(s)n(s) - v_q(s)(n(s))^2$. For each experiment using synthetic data, I tested adding zero-mean Gaussian noise with standard deviation $\sigma \in \{0.1, 1, 10, 100\}$.[2]

For CredNet experiments, my colleagues and I initially generated a CredNet game with 6 players, 6 strategies, and 2644 observations of each payoff, but found that it had particularly high variance; therefore we also generated a second data set with the same players and strategies called CredNet agg., where each of 1000 observations comes from 20 preaggregated runs of the simulator. The true game in my CredNet experiments is always the empirical game constructed using the full set of samples. Algorithms observe the data of this game by drawing randomly without replacement from the set of available observations.

[2]My co-authors examined other settings for game size and noise models for the fixed-sample setting and found little variation in the performance of the bootstrap.

To facilitate comparison of regret values across classes, I applied an affine transformation to rescale each uSym and Cgst payoff matrix to match the range $[0, 100]$, which closely matches the payoff range of the CredNet true game.

5.4 Sequential Classification of Profiles as Nash Equilibria

To test the efficacy of CIBSR at labeling profiles as either δ-Nash equilibria (hereafter referred to as Eq) or not (hereafter referred to as Not-Eq), I measure the frequencies at which the algorithm correctly labels candidate profiles from synthetic or simulated game data. For each game type considered, 1000 trials were run, each trial consisting of labeling a single candidate profile, which may be either Eq or Not-Eq with respect to the true game. For synthetic games, each trial corresponds to a different randomly generated game, while a trial with simulation game data corresponds to a different random ordering of observations of a fixed data set. This means that for the simulated game trials, the set of approximate equilibria of the true game remains fixed across trials.

Selecting profiles randomly for evaluation may be an insufficiently stringent test of the algorithm, since Not-Eq profiles with high regret can be labeled with confidence with very few observations. Furthermore, such profiles would not merit application of statistical tools in practice, as a mixed-strategy profile is typically only of interest if it is believed to be a close approximation of equilibrium. I construct candidate profiles by taking a small number of observations of each profile and computing an equilibrium of the corresponding empirical game. These observations are then passed to the algorithm as Θ_{init}. With this procedure I am able to generate candidate profiles that belong to either class, and Not-Eq instances will frequently be low regret, making correct labeling appropriately difficult.

For the synthetic games, candidates are selected after taking 5 observations of each profile. On each trial the algorithm is parameterized with a regret threshold $\delta = 0.05$ and step size of $x = 5$ observations. Each trial also has an observation cap of 1000 observations per profile, at which point, if the algorithm has not terminated, it labels the point as Eq if the median of the bootstrap distribution is below δ. For the credit network simulator, I tested both the unaggregated and aggregated data sets described in the previous section. Due to relatively high level of noise in the credit network data, candidates were chosen after 100 observations for the unaggregated data, and after 5 observations for the aggregated data.[3]

[3]Since each aggregated observation averages 20 observations, the candidate profiles for both experiments are constructed after the same number of simulations.

Similarly, the algorithm is parameterized with $x = 100$ for the unaggregated data and $x = 5$ for the aggregated data, with observation caps set at the size of the full data set: 2644 and 1000 respectively. Experiments on these games were conducted for $\delta \in 0.05, 0.2, 0.5$, as different settings of δ lead to a different distribution of Eq and Not-Eq instances, and potentially change the difficulty of correct labeling. For all experiments presented here, the algorithm is parameterized with $\gamma = 0.95$. As the algorithm uses a two-sided γ-confidence interval and the final decision depends on only one of the boundaries of the interval, the algorithm terminates with a stated confidence level of 0.975.

Table 5.1 presents the findings from these experiments. "Ground Truth" specifies whether the row refers to instances where the ground truth is Eq or Not-Eq. "#" specifies the number of instances out of the 1000 trials that had the specified ground truth, while "Und.#" gives the number of trials for that setting of ground truth where the algorithm was undecided after reaching the observation cap imposed in the experiment. "Accuracy" lists first the fraction of labelings that were correct for the trials where the algorithm terminated with a confident decision, and second the fraction correct when the algorithm was forced to decide at the observation cap. "Agg. Accuracy" specifies the fraction of labelings that were correct across all 1000 trials, including both Eq and Not-Eq instances. "Observations / Median" and "Observations / Mean" specify the median and mean number of observations taken before the algorithm terminated. In this table, 1.00 signifies that the value is greater than 0.995, while 1 indicates 100% accuracy.

Despite choosing instances so as to increase the difficulty of correct labeling, CIBSR using the bootstrap method to construct confidence intervals delivered high levels of accuracy across all game classes and parameter settings, with the lowest accuracy observed over a full set of trials being 0.922. On synthetic data, CIBSR delivered aggregate accuracy of at least 0.97 across all games and instance types. Note that while CIBSR delivered greater accuracy than the specified 0.975 confidence on the Cgst data, this is because the median number of observations taken prior to terminating for each Cgst setting is 5, with the exception of $\sigma = 100$ where it is 10. In other words, Cgst trials are so easy for the algorithm that it often does not require a second sample. Also of note, on synthetic data Eq instances tended to require fewer observations to make a determination than Not-Eq, with the exception of the highest noise settings. The credit network data proved more difficult, and despite overall high levels of accuracy, accuracy varied considerably between experiments and instance types. In particular, low regret thresholds made the algorithm less likely to label candidates as equilibria; this is reflected in the high accuracy for Not-Eq instances, low accuracy for Eq instances, and higher incidence of inconclusive results for Eq instances. This effect was muted for the CredNet agg. data set, being observed only

Game	δ	Ground Truth	#	Und. #	Accuracy	Agg. Accuracy	Observations	
							Median	Mean
uSym	0.05	Eq	973	0	0.99, N/A	0.989	5	6.27
$\sigma = 0.1$		Not	27	0	0.63, N/A		5	116.11
uSym	0.05	Eq	624	22	0.99, 0.77	0.978	5	144.41
$\sigma = 1$		Not	376	21	0.98, 0.76		47.5	226.66
uSym	0.05	Eq	413	19	0.98, 0.47	0.972	5	113.86
$\sigma = 10$		Not	587	18	1.00, 0.22		15	110.25
uSym	0.05	Eq	240	7	0.97, 1	0.972	30	181.35
$\sigma = 100$		Not	760	11	0.98 0.36		10	68.34
Cgst	0.05	Eq	998	0	1, N/A	1	5	5.23
$\sigma = 0.1$		Not	2	0	1, N/A		397.5	397.5
Cgst	0.05	Eq	911	6	1, 0.67	0.995	5	28.92
$\sigma = 1$		Not	89	1	0.98, 1		50	172.08
Cgst	0.05	Eq	862	0	1.00, N/A	0.996	5	5.99
$\sigma = 10$		Not	138	2	0.99, 0		10	55.07
Cgst	0.05	Eq	610	4	0.99, 0.25	0.984	15	49.11
$\sigma = 100$		Not	390	6	0.98, 0.83		10	53.68
CredNet	0.05	Eq	96	84	0, 0.83	0.974	2644	2347.88
		Not	904	250	1, 1		300	986.43
CredNet	0.2	Eq	642	558	0.75, 0.90	0.922	2644	2485.18
		Not	358	152	1, 0.99		1600	1614.61
CredNet	0.5	Eq	910	512	0.99, 0.99	0.985	2644	2080.81
		Not	90	56	0.91, 1		2644	1976.27
CredNet,	0.05	Eq	428	122	0.86, 0.98	0.953	30	370.26
agg.		Not	572	64	0.99, 1		60	224.38
CredNet,	0.2	Eq	774	156	0.96, 0.95	0.965	85	323.91
agg.		Not	226	18	0.98, 1		280	386.97
CredNet,	0.5	Eq	997	128	0.98, 1	0.983	35	206.98
agg.		Not	3	1	1, 1		130	405

Table 5.1: Sequential classification performance.

for the lowest regret setting. That observing the entire data set for the credit network game was frequently insufficient to have high confidence in declaring that a candidate profile was an δ-equilibrium demonstrates how noisy game simulation can be, necessitating statistical expressions of confidence.

5.5 Sequential Search for δ-Equilibria

The previous experiment demonstrates that the bootstrap method of constructing confidence intervals can successfully be used as a terminating condition for a sequential statistical experiment without drastically biasing inferences drawn at the conclusion of the experiment. Unresolved, however, is whether the bootstrap method can accurately evaluate the outcome of sampling when existing rules of thumbs are employed to guide the sampling process. In this section I present an experiment to evaluate the accuracy of the bootstrap method applied to the conclusion of sampling using two common rules of thumb, as well as CIBEF. I also compare the performance of these rules in terms of average regret and number of observations requested.

The first rule of thumb is to cease sampling when all relevant payoff estimates demonstrate low variability; specifically, the stopping rule labeled SEM will request x further observations be made of each profile in each step until the standard error in mean of each payoff, estimated using the sample variance, is below a specified threshold ξ. The intuition behind this stopping rule is that reliable estimates of payoffs should lead to reliable inferences about the true game. The second rule ceases sampling when the set of equilibria of the empirical game does not change with additional observations. This stopping rule, labeled EQC (for equilibrium comparison), will request x further observations be made for each profile until the sets of empirical game equilibria found in successive steps are equivalent. Distributions used in mixed-strategy equilibria are considered to be equivalent if their Euclidean distance is below some threshold Δ. Both rules of thumb prescribe a stopping point at which equilibria of the empirical game are considered approximate equilibria of the true game, but provide no direct evidence that the profiles they identify are δ-equilibria for any particular δ.

In this experiment, a trial consists of the specified algorithm requesting observations from a synthetic or simulation-based game model until its stopping condition is triggered or an observation cap is reached, at which point it returns one or more equilibrium candidates. If the observation cap is reached, CIBEF returns the equilibrium of the empirical game with the lowest right-hand one-sided confidence bound, while SEM and EQC return all equilibria of the empirical game. For each candidate profile, I record its regret when

played in the true game, as well as the number of observations taken of each profile prior to terminating the sampling procedure.[4] For each game model and algorithm, I ran 1000 trials, with each trial corresponding to a new random game for the synthetic game data, and a new random reordering of the data for the simulated game data. In addition to metrics about the number of observations and regret of the selected profiles, I measure the average fraction of regret of the true games captured by the 95^{th}-percentiles of bootstrapped regret distributions calculated at the termination of each trial. This measure gives an indication of the accuracy of using the bootstrap approach to give a confidence interval at the termination of sampling, when sampling is guided by these sample control algorithms.

EGTA studies often do not allocate the same number of observations to each profile. As a consequence, the precision of payoff estimates may vary across profiles in an empirical game. This varying precision could potentially disrupt the accuracy of statistical conclusions drawn from bootstrapping regret, as regret compares payoffs from multiple profiles. I assess the impact of non-uniform sampling on the accuracy of the stopping rules on the CredNet agg. data set, by using the following sampling method in each step:

1. Let X equal the total number of observations taken in each step, x multiplied by the number of profiles in the game.

2. If this is the first sampling step, assign one observation to each profile and reduce X by the number of profiles in the game.[5]

3. Assign X observations uniformly at random to the profiles of the game.

Since this sampling paradigm will produce varying numbers of observations for each profile, I employ scheme II, which redraws from each profile independently, for bootstrap resampling.

Table 5.2 shows results from the equilibrium search experiments. "Mean Obs." refers to the average number of observations taken of each profile when the algorithm terminated. All trials were conducted with the same step sizes and observation caps as in Section 5.4. Trials labeled "CredNet, agg.; random sampling" employ the sampling approach described in the preceding paragraph, while all others assign x observations to each profile in each step. For the SEM stopping rule, the threshold ξ was set to 1.0, while for the EQC rule, a distance less than $\Delta = 0.01$ was considered sufficient to call two equilibrium candidates identical. For all experiments conducted with the CIBEF stopping rule, δ was set to 0.5 and

[4] All algorithms considered here sample evenly from all profiles in a game's profile space, so "10 observations" refers to 10 observations taken of every value in the payoff matrix.

[5] Each profile is assigned at least one observation so that a mixed-strategy Nash equilibrium can always be found in the empirical game.

γ was set to 0.95. These settings are not particularly optimized for the games considered, but were chosen with reference to the range of payoffs ($[0, 100]$). For EQC, Δ was chosen so as to approximate the ability of a human analyst to distinguish between distributions without computational aid.

These results demonstrate that the bootstrap method of generating confidence intervals for regret may be applied successfully to sequential sampling experiments. The only experiments that resulted in substantial overconfidence on average, that is experiments in which the fraction of true game regrets captured by 95^{th} percentile of the bootstrap regret distribution is less than 95%, were those settings in which sampling typically halted at the first opportunity. For very small numbers of observations, in this case 5, the bootstrap method may suffer from a sample not being representative of the underlying population. For some combinations of game models and stopping rules, the bootstrap generated overly large confidence intervals, with greater than 95% of true game regrets being captured by the confidence interval method. In such cases, requiring bootstrap confidence intervals to be below δ to confirm an equilibrium candidate may rule out more candidates than is warranted by the expressed confidence level; however, when using CIBEF, in contrast to existing methods, an equilibrium meeting the confidence requirements will eventually be found.

In comparing the equilibrium-finding characteristics of the three stopping rules, my experiments show that CIBEF is typically comparable to and often an improvement over existing rules of thumb. For every game model considered, the median regret of the profiles returned by CIBEF was considerably below the approximation threshold of $\delta = 0.5$. Similarly, the mean regret was below the threshold for all but the uniform symmetric game model with the highest variance. This data suggests that in most scenarios when CIBEF misidentifies a profile as an approximate Nash equilibrium it is still likely to have regret close to the threshold, but for high noise settings, profiles that are incorrectly returned may have significant regret. EQC nearly always yielded higher regret profiles than CIBEF, and performed particularly poorly in the synthetic game models with high variance. SEM frequently performed at the same level or better than CIBEF in terms of regret, but may require careful tuning of the ξ parameter in practice, as the algorithm never took a second sample for low noise trials and always took the maximum number of samples for the highest noise trials. In terms of the number of observations taken prior to stopping, CIBEF was similar to SEM for low noise settings, but required many fewer observations for higher noise settings. In contrast, CIBEF required fewer observations than EQC in low noise settings, in part due to EQC requiring two sampling steps prior to terminating, but required slightly more samples in noisier settings.

Game	Rule	Mean Obs.	Mean Regret	Median Regret	.95 Frac.
uSym $\sigma = 0.1$	EQC	17.04	0.0482	0.0011	0.93
	SEM	5	0.0073	0.0014	0.89
	CIBEF	5.08	0.0081	0.0018	0.90
uSym $\sigma = 1$	EQC	17.07	0.0807	0.0107	0.94
	SEM	5.02	0.0730	0.0126	0.90
	CIBEF	5.08	0.0666	0.0101	0.90
uSym $\sigma = 10$	EQC	25.87	0.3771	0.0795	0.95
	SEM	117.32	0.1453	0.0281	0.95
	CIBEF	62.50	0.0716	$5.32e$–7	0.92
uSym $\sigma = 100$	EQC	78.66	2.641	0.6669	0.96
	SEM	1000.0	0.5176	0.1168	0.96
	CIBEF	94.10	0.9257	$1.98e$–6	0.90
Cgst $\sigma = 0.1$	EQC	10.01	0.0006	$1.12e$–7	0.93
	SEM	5	0.0008	$2.20e$–7	0.90
	CIBEF	5	0.0008	$2.19e$–7	0.89
Cgst $\sigma = 1$	EQC	10.04	0.0063	$1.15e$–7	0.92
	SEM	5.01	0.0090	$2.26e$–7	0.89
	CIBEF	5.01	0.0084	$2.31e$–7	0.90
Cgst $\sigma = 10$	EQC	10.83	0.0553	$1.64e$–7	0.91
	SEM	113.14	0.0174	$3.32e$–7	0.96
	CIBEF	15.85	0.0172	$2.76e$–7	0.88
Cgst $\sigma = 100$	EQC	20.06	0.8963	$1.44e$–6	0.98
	SEM	1000.00	0.0771	$2.71e$–6	0.94
	CIBEF	26.94	0.1703	$1.58e$–6	0.97
CredNet, agg.	EQC	66.71	0.0468	$1.03e$–7	0.98
	SEM	116.05	0.0386	$1.96e$–7	0.95
	CIBEF	11.10	0.0295	$1.70e$–7	0.99
CredNet, agg.; random sampling	EQC	72.44	0.0451	$8.61e$–8	0.97
	SEM	119.63	0.0349	$1.82e$–7	0.96
	CIBEF	11.34	0.0370	$1.84e$–6	0.98

Table 5.2: Stopping rule performance.

CIBEF outperformed EQC and SEM in terms of mean regret of returned candidates and the average number of observations taken prior to stopping on the aggregated credit network data when uniform sampling was employed. All rules performed excellently in terms of median regret, meaning that either CIBEF returned fewer non-equilibrium candidates or that the non-equilibrium candidates that it returned were closer approximations to equilibrium than the other two stopping rules. Here, CIBEF may benefit by restricting its output to only candidates that are highly likely to be equilibria, rather than returning multiple candidates that may vary greatly in how well they approximate equilibria, as in EQC or SEM. As such, CIBEF can deliver significant savings in terms of sampling costs when finding only one equilibrium is acceptable. I was unable to gather results for the credit network game with unaggregated data, as this experiment proved very costly, particularly for CIBEF, as it must find equilibria and calculate confidence intervals for them in every sampling step. Though a potential detriment to CIBEF, in real applications of EGTA the cost of sampling will typically outweigh the cost of calculating confidence intervals, thus making the overhead of using CIBEF over EQC negligible.

When using the random sampling paradigm on the credit network data, performance for all algorithms differed only slightly from uniform sampling. All algorithms required slightly more observations to conclude under random sampling. While these additional observations seem to have marginally improved the performance of SEM and EQ in terms of regret, CIBEF performed slightly worse under random sampling. This outcome may be due to the relatively small number of observations that CIBEF takes prior to stopping; with a little over 11 observations of each profile taken on average, random assignment of observations to profiles makes it likely that some profiles have too few observations to provide reliable payoff estimates. The similarity of the outcomes of these trials to the CredNet trials with uniform sampling provides evidence that the accuracy of bootstrap estimates of regret does not hinge strongly on all profiles having the same number of observations, nor on which resampling scheme is employed in the bootstrap.

5.6 Discussion

In this chapter I provide experimental evidence that using the bootstrap is both valid and useful for statistical analysis of empirical games, even when data is gathered and analyzed sequentially.[6] In contrast with classical parametric approaches, the bootstrap enables one

[6] A caveat must be provided: while the techniques I discuss in this chapter performed well in a variety of game settings, there may exist games that I have not tested for which the bootstrap techniques performs poorly.

to make statistical claims without prior knowledge about payoff distributions, overcoming one of the natural challenges of working with data generated through complex simulation. In particular, using the bootstrap to generate confidence intervals for regret allows us to provide evidence as to the likelihood that equilibria identified by sampling from a simulator are indeed equilibria of the true game. Publishing this information alongside equilibrium analysis will provide readers with a better assessment of the evidence supporting the conclusions drawn using EGTA than has been available historically.

I proposed using the bootstrap as part of a stopping rule for two sequential sampling tasks and evaluated the performance of these stopping rules. In the labeling experiment, I found levels of accuracy approximately consistent with stated confidence intervals despite specifically constructing a challenging set of trials. These experiments demonstrated some interesting asymmetries in accuracy at labeling instances of the two classes that may warrant further study. Also of note is that labeling accuracy was slightly lower on the CredNet data set, though this may be due to having to terminate many trials at the observation cap. Further experiments on simulation-based game data would help determine which features of the CredNet data set make labeling with confidence so challenging.

In the equilibrium search experiment, I found evidence that my new algorithm, CIBEF, improves on existing iterative EGTA designs. Not only does the algorithm find equilibrium candidates for which the data suggests high statistical confidence, but often these candidates are found more quickly than with existing rules of thumb. One likely explanation is that CIBEF focuses on identifying the best candidate, and as soon as one profile meets the condition it terminates. Consequently, profiles returned by CIBEF have much lower regret in the true game than those returned by other methods, as fringe candidates are unlikely to be identified early in the process. EQC, the method based on waiting for the set of equilibria to converge, in contrast, may request more sampling simply to eliminate fringe candidates from the set of equilibria, or may cease sampling with such candidates still in the set; either outcome hurts the performance of EQC in the metrics measured here. This explanation also accounts for the relatively weaker performance (in terms of number of observations requested) of the bootstrap at the labeling task. Because regret is a maximum, in order for the expectation of the sampling distribution of regret of a profile to be close to zero it has to be low probability for draws from any deviating distribution to have a higher payoff. As such, profiles with true regret very close to zero should have confidence intervals that converge faster than those with even slightly higher regret.

In the next chapter I revisit an EGTA study that was originally conducted before the development of bootstrap methods for game-theoretic analysis. I apply the bootstrap to calculate confidence intervals on the regret of profiles previously reported as equilibrium. I

also demonstrate how the bootstrap can be applied to estimates of other simulation statistics in the context of an EGTA study.

CHAPTER 6

Application: Equity Premium Estimation in Asset Pricing

Despite the increased availability of economic data, understanding and predicting the behavior of markets remains an intriguing open question. As the recent financial crisis has demonstrated, many economic models rely on assumptions made in the name of tractability that may not be borne out by real world actors (Colander et al., 2009). With computational models, we have the flexibility to relax some of these assumptions, but with greater model complexity comes greater computational costs. Furthermore, such models may be difficult to interpret, limiting potential adoption in mainstream economic and finance research (Leombruni and Richiardi, 2005). EGTA provides a framework for interpreting strategic outcomes from agent-based simulation (also referred to as agent-based modeling, or ABM), addressing this concern, but can have substantially greater costs, as a typical agent-based simulation corresponds to observing a single profile.

In this chapter, I explore the implications of an analytical pricing model in a large, simulated financial market. I conduct EGTA to evaluate whether a prominent analytical pricing model can resolve the famous *equity premium puzzle* (Mehra and Prescott, 1985)—the apparent underpricing of stocks (equity) relative to their expected return as compared to risk-free alternatives. In order to evaluate whether a pricing model resolves this puzzle, I first determine whether or not the pricing model reflects equilibrium play. This is done by implementing the pricing model and logical alternatives as strategies that traders can adopt in a stock market simulator, and finding Nash equilibria of the resulting empirical game. Next, I estimate equity premium at Nash equilibrium. In order for an EGTA study to provide evidence that a pricing model resolves the equity premium puzzle, not only must the corresponding pricing strategy be played in equilibria, but also the magnitude of estimated equity premium should be similar to measurements of equity premium from the real world.

After providing background on agent modeling in finance and the equity premium puz-

zle, I describe my market simulator and how it relates to the analytical model under investigation. In order to conduct EGTA while simulating a large number of traders, I employ hierarchical reduction to build game models, and use the method of control variates (Lavenberg and Welch, 1981) to reduce variance in my payoff and equity premium estimates. I also present a method of employing hierarchical reductions of varying sizes (alluded to in Chapter 3) to reduce simulation expenditure by finding productive portions of profile space with a coarse-grained reduction before committing to more simulation with a finer-grained reduction.

In this chapter I demonstrate one of the key benefits of EGTAOnline: the ability to revisit earlier EGTA studies after the introduction of new methods. The experiments that form the basis of this chapter were originally presented in an issue of the *Journal of Computational and Mathematical Organization Theory* (Cassell and Wellman, 2012). As the data is conveniently stored in EGTAOnline (Chapter 3), I re-examine the published conclusions with the bootstrap methods from Chapter 5. I use the bootstrap to construct regret confidence intervals for the previously identified equilibria. I also present a bootstrap method for computing confidence intervals for equity premium estimates.

6.1 Background: Agent Modeling

Despite a burgeoning literature in agent-based simulation for financial modeling, it remains almost *de rigueur* in mainstream academic finance research to model aggregate behavior in financial markets analytically with a single representative agent. Such agents represent the aggregated preferences and beliefs of all agents in the market, and generally are assumed to hold all of the assets in a market in equilibrium. From these abstractions, pricing and other behavior is derived. Chapman and Polkovnichenko (2009) demonstrated that the lack of agent heterogeneity inherent in the representative agent approach may have important implications, particularly when the agent is not an expected utility maximizer. Specifically, the authors show that adding even one more agent to a market model can qualitatively change the conclusions of an asset pricing study.

Although a two-agent model suffices to demonstrate the weakness of a representative agent model, it may still fail to support the agent heterogeneity required to explain market behavior. Consider a two-agent model, where one agent opts not to trade in equilibrium. Any estimations of macroeconomic variables under this model would then correspond to half of the pool of traders not participating in that market. This level of granularity may inaccurately estimate such variables since no trades are conducted and only half of the market participates in price formation. Additionally, if a two-agent model corresponds to

two different pricing strategies it may not be reasonable to assume that the representative agent opting out in equilibrium implies that no agents of that strategy would participate in the market being modeled.

Because a single representative agent sets prices directly, such models need not consider how prices are actually determined by agent interaction through market mechanisms; however, this can lead to aggregate pricing that is inconsistent with real-world market operation. Consider the agent that has the lowest value among agents trading units of a risky asset through a *continuous double auction* (CDA) (Friedman and Rust, 1993)—an auction in which both buyers and sellers submit prices which are matched continuously. If the market has a large number of orders outstanding then this agent's bids are never seen by the market. His buy price is too low to reach the top of the order book and his sell orders match at the higher market bid. The converse is also true, as an agent who believes the asset is very valuable will ask too much for its sale, but will be able to purchase from other sellers at the lower market ask price. We can thus see that in a thick market, the prices of agents with extreme values for an asset should not factor into calculations of equity premium, since their prices are never seen by the market. With as few as three agents, market microstructure can also affect the allocation of assets, which in turn can affect market pricing, efficiency, and other outcomes.

It might seem the way forward is to construct analytical models with greater numbers of agents; however, as the number of agents increases so does the complexity of analysis. Additionally, the incorporation of market microstructure can cause the complexity of the environment description to increase along with population granularity. Beyond a point, the only feasible computational approach is bottom-up simulation of agent behavior.

Despite limited recognition by mainstream economic journals, ABM approaches have been employed extensively in the social sciences, including finance. LeBaron (2006) surveys the agent-based finance literature, and discusses the motivations and limitations of the approach. ABM has increasingly been employed to explore the impact of public policy decisions on the behavior of market actors. Raberto et al. (2008) use ABM to examine the effects of tight monetary policy with heterogeneous agents. Thurner (2011) conducted an ABM investigation of the relationship between leverage and financial crises, and how the market may respond to some proposed regulations. Gerst et al. (2013) developed an extensive ABM system to study the reaction of domestic actors to changes in international climate policy. It is in a similar vein that I use ABM and EGTA to critically examine an analytical model which may form the basis for policy decisions.

6.2 Ambiguity Aversion and the Equity Premium Puzzle

In standard models of asset pricing, investors demand a higher rate of return as the risk of an asset increases. This is a direct consequence of risk-averse utility: given a choice among two assets with the same expected value, the one with lower risk provides greater expected utility. There are many ways to measure the relative risk of a proposition (Rothschild and Stiglitz, 1970); in my setting below, risk is characterized by the uncertainty in estimating the expected return on the basis of an ambiguous stream of information. An increased rate of return on risky assets as compared to risk-free alternatives is known as the equity premium, as it is a premium demanded by traders for holding equity. The equity premium puzzle refers to the apparent disparity between the observed equity premium and what would be predicted based on standard models of investor risk preferences. If investors were not risk averse, classical economic theory would suggest that the price of stocks would rise until the point where expected return on stock was exactly equal to the expected return on risk-free assets. Since investing in stock has inherent risk, and traders are not in general risk neutral, stock prices should be lower than this point; however, even when accounting for risk aversion, the average return on stock is significantly higher than the return on risk-free treasury bills. Since this phenomenon was identified by Mehra and Prescott (1985), it has received a great deal of attention in finance research literature. There are many proposals for resolving the puzzle, as well as controversies about whether there is a puzzle at all. For example, Weitzman (2007) shows that the puzzle can be explained if the investors' subjective distributions of returns possess heavy tails. DeLong and Magin (2009) provide a survey characterizing the state of knowledge and debate surrounding the U.S. equity premium.

One path taken by economists to explain the equity premium puzzle is to posit forms of non-standard preferences or decision rules. One example is *ambiguity aversion*, a cognitive phenomenon famously identified by Ellsberg, wherein decision makers prefer actions where the chance elements are objectively clear, even at significant sacrifice of expected utility (Halevy, 2007). Epstein and Schneider (2008) (ES) argue that this aversion can be justified in a dynamic context, when information quality is taken into account. The quality of information revealed under an ambiguous prospect may be less useful, in proportion to the degree of ambiguity. To incorporate this ambiguity, ES extend the work of Gilboa and Schmeidler (1989) who demonstrate that a preference for known risks can be captured by worst-case reasoning over a set of non-unique priors. ES show that this worst-case reasoning amounts to an asymmetric response to information depending on its content, since the worst case when receiving positive news is that it is not informative about future dividend

movement, whereas the worst case when receiving negative news is that it is highly informative about future dividend movement. With this rationalization they develop a model of asset pricing with an ambiguity-averse representative agent, and demonstrate that this can explain why even a market of well-diversified investors may still demand compensation for the idiosyncratic risk associated with each asset they hold, since diversification does not reduce ambiguity in the same way that it mitigates risk. This in turn could explain an equity premium, even among savvy investors, and other phenomena of interest.

6.3 Empirical Game Model of Asset Pricing

Taking the ES model as a starting point, I seek to address two questions. First, given the possibility of multiple strategies, is the ambiguity-averse strategy actually present in equilibrium? If traders gain no benefit from being averse to ambiguity, we would expect traders who are averse to ambiguity to be displaced by those who are not, calling into question the validity of modeling the whole market as a single ambiguity-averse trader. Second, in a model with agent heterogeneity and an active market mechanism, does pricing according to the ES model generate significant equity premium?

To answer these questions, I construct an ABM for asset pricing, and perform empirical game-theoretic analysis to evaluate strategy candidates. The full model includes elements that specify the market mechanism, asset definition, and agent strategies, detailed in the following subsections. Figure 6.1 diagrams how the ABM simulator is used to estimate equity premium. In order to estimate the expected value of some target variable V (in this case equity premium), I adopt Nash equilibrium as my solution concept, and weight observations of the variables according to the probability that the pure-strategy profile with which they are associated is played in equilibrium. For a mixed-strategy equilibrium σ^*, this estimate is given by:

$$\mathbb{E}[V_{\sigma^*}] = \sum_{s \in S} \bar{V}_s \Pr(s \mid \sigma^*),$$

where S is the set of pure-strategy profiles in the game, and $\bar{V}_s$ denotes the sample mean of the target variable V when players play s.

6.3.1 Market and Asset Models

Many representative agent models, including ES, give little consideration to market microstructure. Within their analytical framework this seems reasonable, since one agent holds the entire market in equilibrium. This agent would have no one to buy from or sell

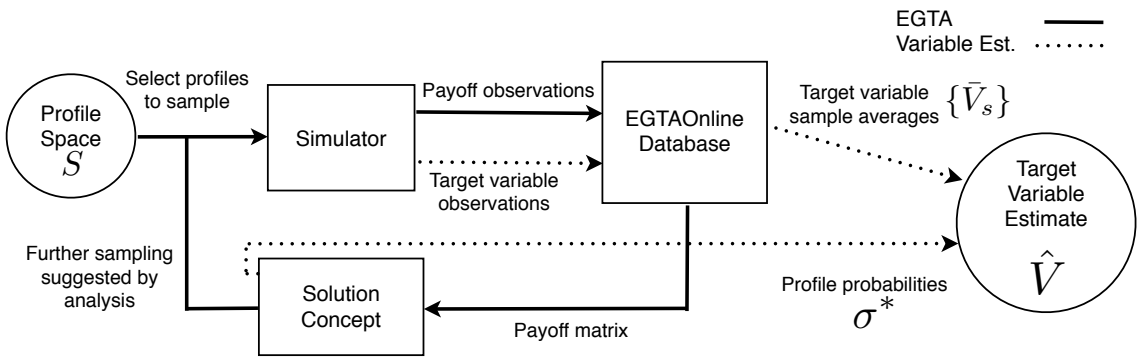


Figure 6.1: Workflow diagram for using EGTA to estimate a non-payoff variables V.

to, so the question of how transactions would occur does not apply. When analyzing price formation via ABM, however, it becomes necessary to specify a mechanism by which the market operates. In many stock markets, agent interaction is mediated through a CDA, which I adopt as the market mechanism for this investigation. Following ES, the model I examine consists of two types of assets, one that is risk-free, offering a fixed return, and one that is risky, offering a variable return. Agents are able to exchange these two assets through the CDA by specifying how much of the risk-free asset they are willing to offer or accept for a share of the risky asset.

6.3.2 Agent Strategy Composition

Given a market mechanism, we must also specify how agents will act within the mechanism to determine prices. The equilibrium bidding strategy for a CDA (or any dynamic market mechanism) in this context is unknown, thus I evaluate a space of candidates to determine an appropriate composition of agent strategies in the model.

My set of strategy candidates starts with the representative agent employed in the ES model. This strategy calculates an asset price using the ES formulation of ambiguity aversion (AA), given its own private information. The strategy bids in a straightforward manner based on that price. To this I add a second strategy candidate, based on a standard Bayesian (B) pricing model, that likewise bids straightforwardly given its price calculation. For the AA strategy to be tenable, it should at minimum be competitive with the natural alternative, B. I also introduce variations on both AA and B that ignore their own risk aversion when calculating prices. This allows me to separately evaluate the advantage or disadvantage to an individual trader of taking risk aversion into account in a similar way to how I have chosen to evaluate the benefits of ambiguity aversion. This set of four strategies is further parameterized by variations to the number of units of the risky asset requested in each or-

der, whether beliefs about the asset's liquidation price are derived from the current market state or the asset's expected value, and a shading parameter that specifies a percentage adjustment that the agents apply to their valuations in order to generate prices. I ultimately explored 20 strategies, curtailing exploration of further parameterizations when it became clear that they offered no strategic advantage.

6.3.3 Estimating the Empirical Game

My market simulation incorporates $N = 60$ symmetric players. With the pool of 20 possible strategies, there are $\binom{79}{60} \approx 8.8 \times 10^{17}$ distinct profiles, taking symmetry into account. Sampling from all of these profiles would be unmanageable; therefore, to contain the profile space further, I employ the aforementioned hierarchical reduction. I start by analyzing a four-player version of the game, where each player selects a strategy to be played by 15 trading agents. Although I have restricted the strategic degrees of freedom (thus sacrificing some fidelity) in the game model, I retain agent heterogeneity in terms of beliefs and preferences at the full granularity of 60 agents. With this reduction, there are only $\binom{23}{4} = 8855$ distinct strategy profiles to sample.

In this profile space, I further reduce my sampling requirements to identify Nash equilibria by use of a guided search that avoids unproductive regions of the game, similar to algorithm introduced by Wellman and Wiedenbeck (2012):

1. Sample the subgame induced by some initial strategy sets.
2. Compute a symmetric mixed-strategy Nash equilibrium (SMSNE) σ of the resulting subgame.
3. Sample from the deviation set of σ in the full game to test for beneficial deviations.
4. If no beneficial deviations are found, return σ; else repeat steps $2 - 4$ by considering the subgame with strategy sets given by $\bigcup_{i \in I} \mathcal{S}(\sigma_i)$, with the addition of a beneficial deviation strategy to the appropriate player's strategy set.

I amend this approach for use with hierarchical reductions of varying granularity, resulting in the Hierarchical-Reduction-Based Search presented as Algorithm 6.1. This search aims to find productive regions of profile space in a coarse-grained reduction, where complete subgames are small relative to the full game. Once productive regions are identified by way of identifying Nash equilibria in the reduced game, the search is refined through using increasingly fine-grained reductions. When payoffs vary smoothly with strategy counts, and approximate SMSNE with small support exist in the true game, this approach can

quickly find equilibria strategy supports and the remaining simulation budget can be spent refining the estimated strategy distributions of equilibrium candidates.

Algorithm 6.1 Hierarchical-Reduction-Based Search

Require: S, the set of all strategies
Require: η, an increasing sequence of divisors of N
  $S_{curr} \leftarrow$ a set initially containing an arbitrarily selected strategy from S
  $S_{old} \leftarrow \emptyset$
  for $n \in \eta$ **do**
    while $S_{curr} \neq S_{old}$ **do**
      Sample all profiles in $\Gamma_{n,S_{curr}}$, the reduced game of size n with strategies S_{curr}
      Find an SMSNE σ of the resulting game model
      $S_{old} \leftarrow \mathcal{S}_S(\sigma)$
      Sample all profiles from $\bigcup_{\hat{\sigma} \in \mathcal{D}(\sigma)} \mathcal{S}(\hat{\sigma})$
      $S_{curr} \leftarrow S_{old} \cup \mathcal{B}_i(\sigma_{-i})$
    end while
  end for

To build the game models, payoff estimates are generated by averaging many observations for each profile. Prior to averaging payoff observations, I apply the method of control variates to reduce variance in payoff estimates. I employ replicator dynamics to find SMSNE of the empirical game, initializing the search from multiple starting points including uniform distributions and distributions weighted towards one strategy. In the experiments I present here, I employ the Hierarchical-Reduction-Based Search with $\eta = (4, 6)$. As discussed in Chapter 3, a considerable fraction of observations gathered for $n = 4$ can be reused for $n = 6$. The six-player game, where each player controls 10 agents in simulation, is what I ultimately employ as the basis of equilibrium and equity premium estimates.

I examine markets where agents are risk averse to varying degrees. Following Mehra and Prescott (1985), I assume that agents' utility for wealth is characterized by *constant relative risk aversion* (CRRA),

$$u(w) = \frac{w^{1-\alpha} - 1}{1 - \alpha}.$$

For $\alpha = 0$ this utility function collapses to $w - 1$, and for $\alpha = 1$ utility is defined by $\lim_{\alpha \to 1} u(w) = \log(w)$. Regardless of whether agents incorporate risk aversion into their pricing strategy, their actual payoffs are calculated with respect to this CRRA utility form. This allows me to examine whether risk-averse pricing is found in equilibrium when agents can strategically ignore their aversion when determining price, and measure the effect that risk aversion has on equity premium.

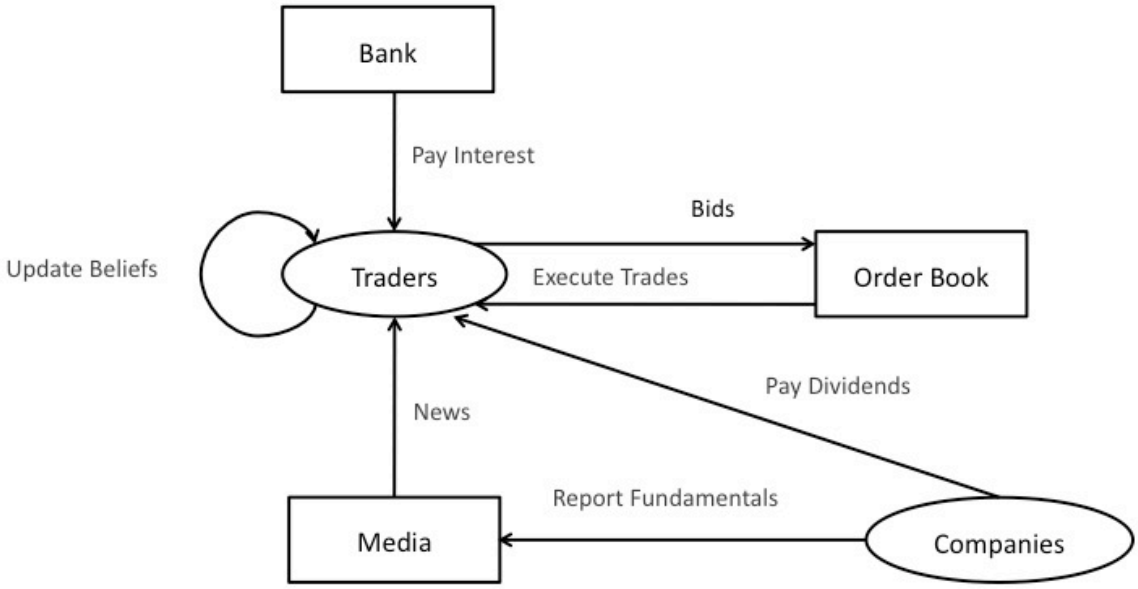


Figure 6.2: Abstract view of market simulation.

6.4 Simulation of Asset Pricing under Ambiguous Information

Figure 6.2 provides an abstract overview of the simulation. In my simulations, in each trading period, the agents—*traders*—are given a piece of news that is partially informative about future dividend payments. Traders use these signals to update their belief about the current value of the risky asset. With updated beliefs, the traders submit some constant number of single-unit limit orders to the order book, where trades are matched using a CDA. Unmatched orders remain on the order book until they either match an incoming order or are replaced by the trader that submitted them. Traders are selected to submit orders to the order book randomly in each period. At the end of each quarter, q, traders receive dividend payments on the risky asset they own, and interest payments on their holdings in the risk-free asset. Each quarter is defined to be some constant number of trading periods. The remainder of this section provides further detail and describes the important assumptions of the simulation model.

6.4.1 Market Conditions

The market consists of two types of assets. A risk-free asset (cash) yields quarterly interest payments at a fixed rate, r. A risky asset (stock) yields quarterly dividend payments d_q,

which fluctuate according to the mean-reverting process used by ES:

$$d_q = \kappa\bar{d} + (1-\kappa)d_{q-1} + \delta_q,$$

where δ_q is a shock to the dividend value at quarter q, $\delta_q \sim Normal(0, \sigma_\delta^2)$, and $\kappa \in (0,1)$ is the degree of reversion toward the mean dividend value, $\bar{d}$. If the dividend value drops below zero, traders are paid no dividend, though future dividend movement continues from the subzero value.

6.4.2 News

News is sent to all traders in every trading period, and every trader receives the same piece of news. A piece of news consists of a signal, s_t, representing intangible information available about the risky asset, and the most recent dividend payment, d_q. The signal is partially informative about the next dividend payment, as

$$s_t = \delta_{q+1} + \varepsilon_t,$$

where δ_{q+1} is the shock in the dividend that will be observed at the next dividend payment, and ε_t is the noise in the signal, $\varepsilon_t \sim Normal(0, \sigma_\varepsilon^2)$, $\sigma_\varepsilon^2 \in [\underline{\sigma}_\varepsilon^2, \bar{\sigma}_\varepsilon^2]$. For my experiments, $[\underline{\sigma}_\varepsilon^2, \bar{\sigma}_\varepsilon^2]$ constrain the space of possible beliefs about the news-generating process. Per ES, traders are assumed to be unable to learn the true news-generating process but can use news to improve their estimates of δ_{q+1} and to update their prices.

6.4.3 Traders

Trader i is given two private values, $\underline{\sigma}_{\varepsilon,i}^2$ and $\bar{\sigma}_{\varepsilon,i}^2$, representing its beliefs about the possible range of variance in the noise added to signals. For trader i, two values are drawn uniformly from $[\underline{\sigma}_\varepsilon^2, \bar{\sigma}_\varepsilon^2]$, taking the smaller value as its lower bound, $\underline{\sigma}_{\varepsilon,i}^2$, and the greater value as its upper bound, $\bar{\sigma}_{\varepsilon,i}^2$. Note that the actual noise parameter may lie outside a trader's belief bounds. Each agent is also assigned a coefficient of risk aversion, α_i, with respect to which it seeks to optimize its CRRA utility. Traders incorporate these private values into their valuation of the risky asset, leading to a market that is heterogeneous in price even when all traders price according to a single strategy.

All pricing strategies considered calculate approximations of a *certainty equivalent price* (CEP), that is, the price at which the trader is indifferent to acquiring an additional

share of the risky asset. To calculate the CEP, traders find the price, p_Δ such that

$$EU(h_c, h_s) = EU(h_c - p_\Delta, h_{s+1}), \tag{6.1}$$

where h_c and h_s are the trader's current holdings in cash and shares respectively. $EU(h_c, h_s)$ is the expected utility function for the trader, given by

$$F_{\delta_{q+1}}\left(-\dot{d}\right)\left(h_c + \frac{h_s p^*}{1+r}\right) + \int_{-\dot{d}}^{\infty}\left[u\left(h_c + \frac{h_s}{1+r}\left(p^* + \dot{d} + x\right)\right)\right] dF_{\delta_{q+1}}(x),$$

where $\dot{d} = \kappa\bar{d} + (1-\kappa)d_t$, the deterministic portion of the dividend, $F_{\delta_{q+1}}(x)$ is the cumulative distribution function for dividend shock, and p^* is an estimate of next period's price for the risky asset. The terms outside of the integral account for the portion of the expected value where the paid dividend is zero.

Since δ_{q+1} is Gaussian, there is no closed-form solution to (6.1), necessitating numerical integration. I replace the upper bound with the mean of δ_{q+1} plus six standard deviations. This upper bound was chosen because preliminary testing showed that it provided approximation errors that were smaller than \$0.005. The source of p^* is one of the strategic parameterizations, with agents either using the traditional Capital Asset Pricing Model (CAPM) price, $\frac{\bar{d}}{r}$, or the current midpoint of the bid-ask spread. This distinction is roughly analogous to that of fundamental and technical traders. Finally, p_Δ is calculated through a fixed-point iteration. A similar approach is used to calculate sell prices, replacing h_{s+1} with h_{s-1}. The next two subsections describe how the two families of pricing strategies apply trader-specific private values to modify their CEP estimate.

6.4.3.1 Ambiguity-Averse Pricing

AA traders believe that any variance in the noise in the range $[\underline{\sigma}^2_{\varepsilon,i}, \bar{\sigma}^2_{\varepsilon,i}]$ could describe the true news generating process. Therefore, AA traders maintain a set of priors over this distribution. When ambiguity-averse traders receive a piece of news, they update their beliefs of the distribution of next period dividend shock, $F_{\delta_{q+1}}$, using Bayesian updating over their set of priors. Thus trader i's set of posterior beliefs about δ_{q+1} after receiving s_t are summarized as follows:

$$\begin{aligned}\{\delta_{q+1} \mid s_t \sim Normal(\mu(\sigma^2_\varepsilon), \sigma^2(\sigma^2_\varepsilon)) : \sigma^2_\varepsilon \in [\underline{\sigma}^2_{\varepsilon,i}, \bar{\sigma}^2_{\varepsilon,i}]\} \\ \mu(\sigma^2_\varepsilon) = [1-\gamma(\sigma^2_\varepsilon)]\mu_{t-1} + \gamma(\sigma^2_\varepsilon)s_t \\ \sigma^2(\sigma^2_\varepsilon) = [1-\gamma(\sigma^2_\varepsilon)]^2\sigma^2_{t-1} + [\gamma(\sigma^2_\varepsilon)]^2\sigma^2_\varepsilon\end{aligned}$$

where $\gamma(\sigma_\varepsilon^2) = \frac{cov(s,\delta)}{var(s)} = \frac{\sigma_\delta^2}{\sigma_\delta^2+\sigma_\varepsilon^2}$, and μ_{t-1} and σ_{t-1}^2 are the posterior mean and variance of $\delta_{q+1} \mid s_{t-1}$ respectively. After updating their beliefs, AA traders price according to the posterior distribution that minimizes the expected value of their holdings.

6.4.3.2 Bayesian Pricing

B traders treat all news as equally informative and thus respond symmetrically to news regardless of its content. For my experiments, B trader i is assigned $\bar{\sigma}_{\varepsilon,i}^2$ and $\underline{\sigma}_{\varepsilon,i}^2$, but uses the average of these values to form $\sigma_{\varepsilon,i}^2$, upon which their prior belief is based. Thus B traders hold only a single prior for performing Bayesian updating, as compared to the range of priors maintained by AA.

6.5 Experiments

I conducted EGTA and estimated equity premium for several market configurations. Since the space of market parameterizations is continuous in multiple dimensions, such an investigation cannot hope to be exhaustive. Instead I consider a small set of reasonable and representative settings of market parameters.

Some guidance for parameter setting can be taken from the literature. For example, Exley et al. (2004) cover various measures of mean reversion in finance fairly comprehensively. The distribution for α, the coefficient of risk aversion, was chosen to be consistent with the survey of behavioral studies presented by Mehra and Prescott (1985). The range of possible ambiguity in news, however, may be impossible to estimate from a finite sample, and thus there is little guidance available for setting this parameter. This difficulty in determining the news generating process is one of the underpinnings of the ES model, so the lack of established parameter settings is not surprising. In the absence of authoritative prescriptions, I chose parameters to create potentially interesting scenarios. Table 6.1 presents the base market configuration considered, labeled as BASE throughout. For BASE, I chose an environment in which the variance of the dividend movement and the added noise were equal. In this scenario, the probability of a large change in the next-period dividend payment is small, but, for any single piece of news, it is difficult for agents to interpret how much of the news signal is noise.

Table 6.2 summarizes other configurations I examined, specified in terms of their deviation from BASE. Configuration HNV (higher noise variance) increases the ambiguity in the signal stream by increasing the variance in the noise and simultaneously increasing the range from which players' priors can be drawn. As this range increases, AA and B

Parameter	Setting
Q	40
κ	0.05
$[\underline{\sigma}^2_\varepsilon, \bar{\sigma}^2_\varepsilon]$	$[0.0, 0.01]$
σ^2_ε	0.0025
σ^2_δ	0.0025
$\bar{d}$	2.0
r	0.01
cash endowment	10000
stock endowment	50
trading periods per quarter	10
α	$\sim Normal(2.0, 0.5)$

Table 6.1: Base market configuration for experiments.

pricing strategies diverge, since AA traders can now have greater asymmetry in their response to news based on its content. Additionally, dividend movement may be difficult to predict accurately in HNV as the variance in the noise is much larger than the variance in dividend movement. Configuration HDV (higher dividend variance) increases the variance in the dividend movement, implicitly making signals more informative while simultaneously making the asset riskier. Configuration HDVHK (higher dividend variance, higher kappa) is designed to produce spiky movement in dividend payments that nevertheless reverts quickly to the mean. This could make predicting the next period's dividend payment difficult, but makes the risky asset's value more predictable over longer horizons, to the peril of agents that overreact to dividend movement. The ILLIQUID configuration attempts to model a market with infrequent news and trades occurring, as agents only have one trading period before dividends are paid. This scenario seems in some sense contrived, but may be analogous to markets that are very thin. For such markets, trades occur infrequently, and the news coverage of assets in these markets may be less than for popular stocks. Longstaff (2006) argues that many important classes of assets are illiquid in this sense.

6.5.1 Equilibrium Analysis

The first test of a representative agent pricing model is whether or not we would expect any agents to adopt that strategy in equilibrium. Table 6.3 presents the approximate symmetric

Name	Modifications from BASE
HNV	$\sigma_\varepsilon^2 = 0.01$, $[\underline{\sigma}_\varepsilon^2, \bar{\sigma}_\varepsilon^2] = [0.0, 0.04]$
HDV	$\sigma_\delta^2 = 0.04$
HDVHK	$\sigma_\delta^2 = 0.04$, $\kappa = 0.2$
ILLIQUID	trading periods per quarter $= 1$

Table 6.2: Variant market configurations tested.

Configuration	Equilibrium	95% Regret
BASE	BNF2-0: 0.67, BRT1-0: 0.19, BNT1-2: 0.09, BRF1-0: 0.06	106.93
HNV	BNF1-0: 0.74, BRF1-0: 0.15, ARF1-0: 0.06, BRT1-2: 0.03, BRT1-0: 0.01	200.64
HDV	BRF1-0: 1.0	1075.99
HDVHK	BNF1-0: 0.56, BNT1-0: 0.23, ANF1-0: 0.09, BRT1-2: 0.08, ARF1-0: 0.04	98.28
ILLIQUID	BNF1-0: 0.63, BRT1-0: 0.37	178.87

Table 6.3: Symmetric mixed equilibria found in each of the market configurations.

Nash equilibrium identified through replicator dynamics for each market configuration[1] under the six-player reduction, as found by the original study. In the table, an equilibrium is specified by a map from strategies in support to their probability of play. Each strategy in the table is abbreviated with the pattern [A = AA, B = B][R = priced with risk aversion, N = priced without][T = estimated p^* with CAPM, F = estimated p^* based on current price][number of shares requested per order]-[percent bid shaded down by buyers and up by sellers]. The column headed "95% Regret" gives the right-hand bound of a one-sided 95% bootstrap confidence interval on regret for the equilibrium candidate, computed as described in Section 5.1. Note that while the equilibrium candidates were identified in the games constructed using the method of control variates to reduce variance, the regret confidence intervals were computed using the raw data.

None of the equilibria identified featured AA strategies played with probability greater than 0.13. There does not appear any particular pattern regarding whether or not to incorporate risk aversion into pricing, despite the fact that payoff calculations at the end of

[1]Strategies where the agents submitted orders for more than one share were examined only under the BASE configuration, as inspection of payoffs in this game suggested that such a simple change to order quantity did not meaningfully affect outcomes.

simulation explicitly include risk aversion. For all equilibria, using the current price as a proxy for the liquidation price of an asset is played with significantly higher probability than using the CAPM price as the estimate, though both parameterizations are found in all equilibria other than HDV. Though shading is played in three of the equilibria, it is never played with probability greater than 0.09, and the highest level of shading tested (5%) is not played in any equilibrium. One possible explanation for the lack of shading in equilibrium is that with a market that is thick around the median price, traders engaging in aggressive shading never find matching prices, since their prices deviate significantly from the median price. Such strategies are thus unable to take advantage of mutually beneficial trades that arise due to agents optimizing for differing risk preferences. Overall, I found considerable heterogeneity and variation in equilibrium composition across environments. This in and of itself argues against the a priori imposition of representative agent strategies for models of asset pricing in environments similar to those studied here.

When evaluating the statistical evidence for these equilibrium candidates in the raw data, I find relatively low estimates for the 95^{th} percentile of regret for most candidates. Payoffs in these games are all in the range $\$27,000$–$\$33,000$, and as such, a regret of 100 corresponds to approximately 0.33% of the available payoff. For the HDV setting, however, the estimated 95^{th} percentile for regret is an order of magnitude higher, and given the range of payoffs, cannot be considered a good approximation of equilibrium. For this setting, the conclusions derived using control variates disagree with those of the bootstrap. One possible explanation for this is that the data may contain one or more extreme outliers; while applying control variates could adjust these outliers closer to the expected value, the bootstrap approach incorporates those outliers into its estimate. If the data set does contain such outliers, further sampling could bring the bootstrap results into accord with those derived using control variates. It is also possible that applying control variates in this case led me to halt the original study with insufficient data to support my conclusions, and that further sampling would provide further evidence that the equilibrium candidate was a poor approximation to Nash equilibrium.

6.5.2 Equity Premium Estimation

For each market configuration I constructed equity premium estimates by weighting control-variate-adjusted observations from simulation by how likely their profile is to occur in equilibrium. Within a simulation instance the market microstructure serves as the method of price aggregation, with $\bar{p}_q$ equal to average price at which transactions occurred in quarter q. Equity premium within a quarter, P_q, then, is calculated as the difference

between the return on the risky asset and the return on the risk-free asset over the quarter,

$$P_q = \frac{\bar{p}_q + d_q}{\bar{p}_{q-1}} - (1 + r).$$

This measure compares the return on \$1 invested in the risky asset to \$1 invested in the risk-free asset over the period. In the rare event that no trades take place in a given quarter, no observations are made of equity premium.

In addition to the previously published equity premium estimates, I also devised the following bootstrap method and used it to construct confidence intervals for equity premia:

1. Let b be the total number of equity premium observations over all pure-strategy profiles in $\mathcal{S}(\sigma)$ for a candidate equilibrium σ.

2. Construct a resample of equity premium observations $\hat{\Theta}$ of size b, with each observation chosen in the following way:

   (a) Randomly select a pure-strategy profile s to sample with probability distributed according to σ.

   (b) Draw an equity premium observation uniformly at random with replacement from the set of observations associated with s.

3. Calculate the mean of $\hat{\Theta}$

4. Repeat steps 2 and 3 1000 times to construct the estimated sampling distribution of equity premium for σ.

5. Take the 95% confidence interval to be the 2.5% and 97.5% quantiles of this distribution.

This method estimates the sampling distribution for mean equity premium by weighting the likelihood of drawing a particular observation according to the probability that the observation's pure-strategy profile is realized under a target strategy mixture, here a Nash equilibrium. In contrast to the bootstrap methods from Chapter 5, this method does not build a bootstrap game from the full data set, but instead constructs equity premium resample populations directly. The estimates constructed with this method also differ in meaning from those constructed using the method described in Section 6.3. The sampling distribution estimated by this method corresponds to drawing b observations of σ, where each observation corresponds to the realization of a pure-strategy profile, and computing the

Configuration	Reported Equity Premium	Std. Dev.	95% Bootstrap CI
BASE	$1.52e$–4	9.813–4	$[1.19e$–$4, 2.10e$–$4]$
HNV	$-3.69e$–5	0.00106	$[-2.28e$–$5, 3.87e$–$5]$
HDV	0.00381	0.00963	$[0.00232, 0.00641]$
HDVHK	$3.11e$–4	0.00146	$[2.80e$–$4, 3.49e$–$4]$
ILLIQUID	$4.67e$–6	$6.92e$–4	$[-3.47e$–$5, 1.20e$–$4]$

Table 6.4: Equity premium statistics for each market configuration.

mean of this observation set. As such, it combines many sample populations into an estimated distribution. The method introduced in Section 6.3, in contrast, is a weighted mean of means, a combination of point estimates into another point estimate.

Table 6.4 presents the equity premia calculations for the discovered equilibria. In this table, "Reported Equity Premium" is the equity premium estimates presented in the previous publication, computed using control-variate-adjusted observations. "Std. Dev." refers to the sample standard deviation in the raw data, and "95% Bootstrap CI" is the two-sided confidence interval for equity premium constructed using the bootstrap on the raw data. As with the equilibrium analysis, the bootstrap confidence intervals were constructed using the raw observation data. The highest equity premium estimate for any equilibrium composition was less than 0.4%, with this value being observed for HDV, the setting for which the quality of equilibrium approximation was called into question by the bootstrap statistics. Comparing my estimates to the equity premium estimates from U.S. stock market data presented by Fama and French (2002), between 2.55 and 7.43%, we see that none of these constitute a significant equity premium, even if one uses the right-hand side of the 95% confidence interval as the point of comparison.

For the most part, the equity premium estimates derived using the weighted mean with control-variate adjusted observations are consistent with the bootstrap confidence intervals generated from the raw data. One notable exception is for the HNV setting, where the original equity premium estimate is below the lower bound of the confidence interval. As with the equilibrium analysis, it is possible for outliers to be incorporated into bootstrap resamples that would be smoothed away by applying control variates; however, given that the control variate estimate is more negative than the bootstrap estimate, this explanation would only make sense if the sample population was skewed right. Computing the skew of this sample distribution does show a small skew right of 0.028. It is also possible that this discrepancy arises from differences in the way the two equity premia estimation methods incorporate data. The weighted mean of means approach treats observations from two

different profiles with the same weight as equally likely, even when there are a different number of observations taken for each. In contrast, the bootstrap method gives a higher probability of drawing a particular observation from a smaller set of observations than from a larger set, all else being equal. Consequently, if an outlier is associated with a profile with small number of observations, this observation will contribute more to the bootstrap estimates than to the weighted mean of means. In this data set, some profiles have up to twice as many observations as other profiles, and so this distinction between the two methods may be meaningful.

To assess whether the absence of ambiguity-averse pricing (or presence in low numbers) in equilibrium is responsible for the relative paucity of equity premium, I measured the equity premia in pure-strategy profiles where a single ambiguity-averse pricing strategy was played by all agents. These results are presented in Table 6.5. No combination of market configuration and pricing strategy demonstrated equity premium above 0.003. In concert with equilibrium equity premium estimates above, two factors seem to be strongly correlated with equity premium. In both result sets, the highest equity premia were observed in market settings with increased variance in the dividend, HDV and HDVHK. This effect may be the result of orders remaining in the order book until they are matched or replaced. When the dividend moves upward, some stale sell orders may be exploited to get the risky asset at a discount, increasing the equity premium measured in the next period. When the dividend moves downward, the equity premium measured in the current period can be slightly inflated, as the mean price measured in the period may include trades with stale buy orders. The other factor correlated with equity premium is whether traders price using CAPM price or the current price as their estimate of the liquidation price of the risky asset. For HDV, HDVHK, and ILLIQUID, strategies incorporating the current price (denoted F in the strategy label) lead to higher equity premia than those using CAPM. In the equilibrium equity premium estimates above, the two highest equity premia are associated with equilibria in which 100% (HDV) and 69% (HDVHK) of traders use the current price as part of their pricing strategy. Though the ultimate cause of equity premium is still unresolved, these results suggest that other factors play a greater role than ambiguity aversion.

6.6 Discussion

Estimation of market variables derived through agent interaction is fraught with complexity. Heterogeneous beliefs and pricing strategies combine through a market mechanism to determine trade outcomes, in a manner generally not amenable to closed-form characterization. Although details of strategies and market microstructure can potentially shape

Configuration	Strategy	Mean Equity Premium	95% Bootstrap CI
BASE	ANF1-0	$-1.47e$–4	$[-4.31e$–4, $1.45e$–4$]$
	ANT1-0	$-1.54e$–4	$[-3.82e$–4, $9.71e$–5$]$
	ARF1-0	$-4.22e$–5	$[-1.38e$–4, $4.66e$–5$]$
	ART1-0	$-1.13e$–5	$[-2.02e$–4, $1.94e$–4$]$
HDV	ANF1-0	0.00208	$[0.00113, 0.00308]$
	ANT1-0	$9.12e$–4	$[3.84e$–5, $0.00179]$
	ARF1-0	0.00247	$[9.79e$–4, $0.00412]$
	ART1-0	$3.06e$–4	$[-3.73e$–4, $9.55e$–4$]$
HNV	ANF1-0	$-4.08e$–4	$[-8.71e$–4, $-2.56e$–6$]$
	ANT1-0	$2.42e$–4	$[6.26e$–5, $4.13e$–4$]$
	ARF1-0	$-1.54e$–4	$[-6.57e$–4, $3.97e$–4$]$
	ART1-0	$2.67e$–4	$[1.03e$–4, $4.24e$–4$]$
HDVHK	ANF1-0	0.00103	$[3.83e$–4, $0.00165]$
	ANT1-0	$-1.32e$–4	$[-3.41e$–4, $7.47e$–5$]$
	ARF1-0	$8.62e$–4	$[1.70e$–5, $0.00168]$
	ART1-0	$-1.64e$–4	$[-3.78e$–4, $5.90e$–5$]$
ILLIQUID	ANF1-0	$1.96e$–4	$[-9.27e$–5, $4.60e$–4$]$
	ANT1-0	$-3.36e$–6	$[-2.23e$–4, $2.10e$–4$]$
	ARF1-0	$1.18e$–4	$[-1.92e$–4, $3.97e$–4$]$
	ART1-0	$-1.51e$–4	$[-3.39e$–4, $5.26e$–5$]$

Table 6.5: Equity premium measurements for each market configuration when a single ambiguity-averse pricing strategy is played.

results, these aspects of real-world market interactions are often ignored by modelers in favor of analytical tractability. Representative agent models in particular can often yield strong conclusions, by identifying a single agent with an equilibrium outcome. Whether this abstraction is reasonable, however, depends on how much heterogeneity and market microstructure actually drive the particular environment under study.

In this chapter I presented an EGTA study of simulated stock pricing behavior, employing scaling techniques discussed in this thesis and elsewhere to analyze a simulated market with 60 traders. In doing so, I demonstrated how EGTA can be used for investigating market dynamics with greater fidelity than is afforded through traditional analytical means. For abstract model results to be convincing as a reflection of the real world, arguably they should first be verifiable in worlds intermediate in complexity between the abstract model and reality. The agent-based modeling framework is particularly suitable for introducing agent heterogeneity and capturing detailed interaction mechanisms, and through EGTA, I am able to reason about outcomes at strategic equilibria. In this chapter, I presented two methods for estimating market variables under a mixed-strategy Nash equilibrium, using equilibrium probabilities to weight variable observations made while simulating pure-strategy profiles. The bootstrap approach has the further benefit of estimating a sampling distribution, rather than a single point estimate, making possible the construction of confidence intervals for these variables. I also proposed a new sample control algorithm for use with hierarchical reductions that employs increasingly fine-grained reductions to allocate fewer observations to unproductive regions of profile space.

The EGTA application in this chapter investigated a model of ambiguity aversion presented by Epstein and Schneider. My game-theoretic analysis suggests that the ES model of ambiguity-averse pricing can play a role in equilibrium strategy for certain market configurations, but it is not in general a sufficient representation of trader behavior in my simulated markets. Neither ambiguity-averse pricing nor moderate levels of risk aversion were found to generate significant equity premia in expectation for markets at strategic equilibrium. The greatest equity premium I computed was considerably less than even conservative estimates of the equity premium in the U.S. stock market over the period from 1951–2000. These experiments lend further support to the argument presented by Chapman and Polkovnichenko (2009), that heterogeneity and market microstructure can produce results at substantial variance from representative agent models. This is not particular to the AA pricing strategy, as indeed even a standard Bayesian risk-averse representative agent would have predicted different outcomes than what I observed, as price aggregation with a market mechanism such as a CDA differs from that of a representative agent model. However, the possibility remains open that an expanded model could exhibit different pricing

behavior. For example, it might be relevant to allow for the possibility of the issuer of the risky asset going bankrupt, and thus make the asset permanently worthless, or account for the fact that traders generally have different horizons that shape their expectations about liquidation price. Whether such variants or others would entail qualitatively different pricing strategies or support significant equity premia remain questions for future work.

CHAPTER 7

Conclusion

In this thesis, I presented new software systems and methods for conducting empirical game-theoretic analysis on large, finite games. In so doing, my aim is to alleviate size and complexity restrictions that have in the past dominated the design of EGTA studies. The design of simulators and experiments under the EGTA framework should be guided not by what can be accomplished on a single computer, but instead through consideration of what features of the strategic scenario are important to include in the model. Though conducting EGTA studies remains computationally expensive, the size and variety of EGTA studies conducted by my lab in the last several years is considerable, thanks in part to work described in this thesis. In this concluding chapter, I summarize my contributions, discuss future avenues for scaling the EGTA methodology, and conclude with a few final remarks.

7.1 Contributions

7.1.1 Software Systems and Methods

One insight I gained during the early stages of my graduate career is that a major hurdle to designing high quality simulation-based experiments is how much tedious, non-research work is involved. Modeling can be very fun and rewarding, but moving data between machines, figuring out how to schedule work on a compute cluster, and ensuring that you are gathering all of the data that you require are not tasks that many people relish. Making matters worse, the larger the game, the more of this non-research work there is to do. My approach to scaling is to develop systems that can do such work automatically and efficiently.

Figure 7.1 is an idealized view of how the systems and methods I presented could be combined to automate the non-modeling aspects of EGTA (discussed in further detail below). In Chapter 3, I introduce EGTAOnline, software infrastructure for managing distributed game simulation. EGTAOnline enables users to take advantage of simulation-level

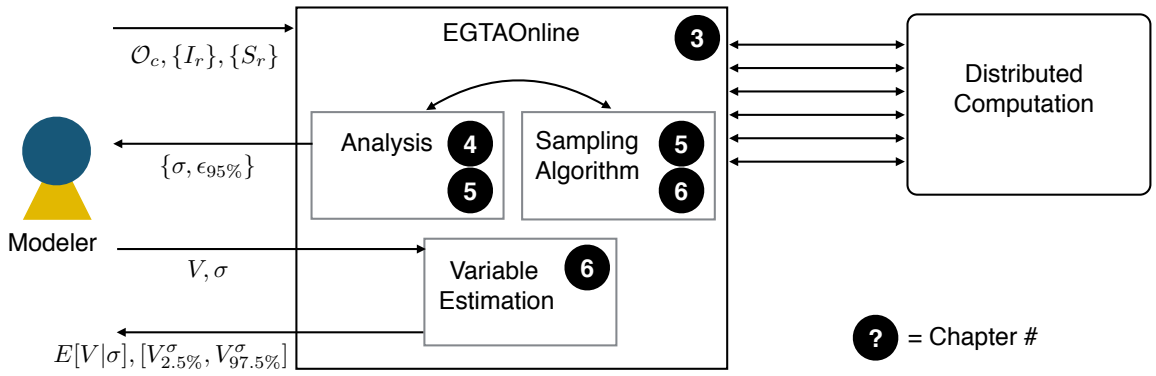


Figure 7.1: Idealized automated empirical game-theoretic analysis.

parallelism without having to learn how to schedule simulation to run on the university compute cluster. I had previously observed that many studies, my own included, had eschewed using such resources, because using the cluster was reputed to have a steep learning curve. EGTAOnline also manages the tremendous volume of data that large EGTA studies generate, and makes this data accessible primarily through game representations that are suitable for analysis. Data that is incompatible for game-theoretic analysis cannot accidentally be combined into a single game, simplifying the process of conducting multiple experiments simultaneously. EGTAOnline also encourages the study of large games through extensive support for role-symmetric game representations, enabling modelers to include exactly as much symmetry in their model as their scenario actually calls for, and by making it easy to schedule simulation for game reductions—abstractions that reduce the space of profiles to sample.

The most common way of scheduling simulation with EGTAOnline today is through selecting a supported scheduling pattern in the web interface and supplying it with details to fill out the space of profiles to sample. As EGTA is often conducted iteratively, changing the profile space to be scheduled on the basis of interim analysis is common, and is most often done manually. The functionality exists in EGTAOnline for users to automate this process with a scheduling script, and this functionality was exploited by Wellman et al. (2013); however, most users of EGTAOnline do not use scheduling scripts currently, preferring the comfort of interacting with EGTAOnline's web interface. One way to alleviate this pain point is by incorporating more sophisticated sampling algorithms into EGTAOnline.

A consequence of automating the sampling process within EGTAOnline is that we have the opportunity to encourage best practices without putting any additional burden on the user. For example, most publications on EGTA studies, including the two applications that I cover in this thesis, have presented results without statistical quantification of confidence. Instead, analysts use ad hoc methods to establish to themselves that the results are trustwor-

thy, such as employing control variates to reduce variance in payoff estimates, or sampling until the set of equilibria of the empirical game does not change with the addition of more observations. With the introduction of the bootstrap to the EGTA toolbox (Wiedenbeck et al., 2014), we can now make statements of confidence about the true-game regret of a profile on the basis of empirical observations. Using this technique, in Chapter 5 I present new algorithms for sampling sequentially to either determine whether a given profile is a δ-Nash equilibrium, or to identify a profile for which we have high confidence that its regret is below a pre-specified δ. I demonstrate that using the bootstrap in this way, we can not only statistically quantify the likelihood of a profile being a δ-Nash equilibrium, but can often reach high levels of confidence with fewer observations than with ad hoc methods. Incorporating such algorithms into EGTAOnline as another scheduling pattern that users can select in the web interface would likely increase the prevalence of statistical confidence information in EGTA studies, as well as reduce excess sampling.

The bootstrap algorithms I introduced always request a fixed number of observations of all profiles in a set of interest in each sampling step. Other sequential sampling algorithms, such as the profile exploration algorithms examined by Jordan et al. (2008), select profiles for observation on the basis of interim game-theoretic analysis. To implement such algorithms within EGTAOnline, the system should be able to supply this analysis in an efficient and automated way. In Chapter 4 I discuss conducting game-theoretic analysis in the database as a step in this direction. I present a schema, compatible with the role-symmetric representation of EGTAOnline, for storing game data in a relational database. With this schema, I construct SQL implementations of common game-theoretic operations. This work is motivated by observing unnecessary costs associated with using existing game-theoretic analysis software. In particular, I find that the cost of building an in-memory object representation of a game in Gambit, a prominent analysis package, significantly outweighs the cost of identifying pure-strategy Nash equilibria. The measurements I present in Chapter 4 actually underestimate the cost of using Gambit, as first users must coerce their game data into the Gambit file format. By moving analysis to the database, the costs associated with moving between representations are eliminated, and I find that this provides a net performance benefit despite the overhead of orchestrating analysis through a database management system.

In Figure 7.1 I diagram how, by combining these elements, the modeler's contribution becomes primarily research tasks, such as defining the model of interest, and choosing what form of analysis they wish to conduct. In this idealized the system, the modeler would supply EGTAOnline with a simulator instance $\mathcal{O}_c$, partition of players into roles $\{I_r\}$, and a set of strategies available to each role $\{S_r\}$, and indicate that they want to iden-

tify mixed-strategy Nash equilibria of the true game. From this point, EGTAOnline would choose a sampling algorithm to schedule simulation to a compute cluster or other form of distributed computation. In conjunction with the analysis module, which would conduct game-theoretic and statistical analysis, EGTAOnline could synthesize the results of simulation into a set of equilibrium candidates and their associated 95% regret confidence levels, and return this set to the user. Another use case is that the modeler may wish the estimate a variable V that results from agent interaction, such as the clearing price of a simulated market, given that strategy adoption is distributed according to some mixed-strategy profile σ. In this case, EGTAOnline could use the bootstrap estimation approach discussed in Chapter 6, and return to the user not only an estimate of V, but also a confidence interval for V, under the supplied distribution. In contrast to the tedious and complex simulation and data management of the past, here the modeler is focused almost exclusively on core research tasks. Given the relative expense of human labor as compared to computer labor, this shift in workload from modeler to computer could net significant savings and lead to increased research productivity.

7.1.2 Applications

I bookend this thesis with two EGTA applications. The first, presented in Chapter 2, is an example of an EGTA study conducted without the benefit of scaling techniques. I explore the question of choosing a wireless network to connect to when faced with multiple options in a game setting, incorporating two emerging wireless technologies: the ability to connect to more than one access point simultaneously, and the ability to probe access points to assess their congestion. In this study, my game model consisted of a dynamic game in which six symmetric players choose among six initially identical access points to use, with congestion on each access point persisting between rounds. I find that the Hedge algorithm (Kleinberg et al., 2009), which assigns work randomly weighted according to estimates of congestion at each access point, is played with high probability in equilibrium when players are informed of the congestion at each access point at the start of each round. When instead agents can gather information only through probing or using an access point, the decision-theoretic strategy, in which agents choose the access point that appears to them to be the least congested, becomes the majority component of equilibrium. Interestingly, reducing the information that is freely available to agents actually improved outcomes in this study, as agents were able to find better assignments of work without the risk of random assignment through the decision-theoretic strategy.

In contrast to the small access point selection study, the equity premium study in Chap-

ter 6 exploits scaling techniques proposed in this thesis, as well as existing scaling techniques, to analyze a simulated market with 60 traders. In this study, I examine the ability of a prominent analytical pricing model to resolve the equity premium puzzle—the apparent underpricing of stocks relative to risk-free alternatives, even when accounting for risk aversion. I conducted EGTA to analyze this problem by implementing the pricing model as a strategy for agents to adopt in simulation, and adding to the strategy set the logical alternative to this pricing model, as well parameterizations of both base pricing strategies. I find that in equilibrium, the original pricing model is not played with high probability, and that there is little equity premium enforced at equilibrium. Despite the fact that these experiments were originally conducted several years ago, I was able to reexamine the data when writing Chapter 6 as it is stored in the EGTAOnline database. To the original analysis I added bootstrap confidence intervals for the regret of previously reported equilibria. In doing so, I found one setting where the published equilibrium had a 95% confidence interval significantly exceeding what would reasonably be classified as an approximate equilibrium. This finding underscores the challenge of interpreting analysis when data has been gathered in an ad hoc manner and confidence information is not supplied. I also computed bootstrap confidence intervals for estimated equity premia, with results that were largely in line with my previously reported estimates.

7.2 Future Work

In each chapter I point out areas that merit additional attention, but some ideas for future work are not strongly associated with any particular chapter. The most immediate future work suggested by this thesis is actually combining the elements herein, as suggested in Figure 7.1. At present, EGTAOnline does not automate any analysis, and accordingly, cannot automatically adjust schedulers on the basis of this analysis. Additionally, no profile exploration algorithms have yet been proposed that incorporate the statistical information derived from the bootstrap; it remains an open question as to the optimal sequential sampling algorithm for searching a large profile space and identifying δ-Nash equilibria with high statistical confidence. Such an algorithm would also ideally account for the availability of parallel computation, and, since computation is a finite resource, prioritize sampling across several experiments running on EGTAOnline concurrently. It is unclear whether a small set of sampling algorithms can hope to address the needs of all EGTA practitioners.

One approach to scaling EGTA that I have discussed very little is using a small set of profiles to approximate a larger game. The hierarchical and deviation-preserving reduction are two methods for this, wherein a simulator with a large number of agents is mapped to

a smaller game on which analysis is conducted. Another avenue, however, is using machine learning to estimate utility functions from data. Vorobeychik et al. (2007) presented methods for learning utility functions in infinite games, but it is not clear that these methods generalize, particularly to arbitrary finite games. Such methods have the promise of dramatically reducing the fraction of the profile space that must be observed to analyze games.

7.3 Final Remarks

EGTA provides a framework for analyzing incentive properties of complex social systems through agent-based simulation. This analysis typically focuses on comparing outcomes from a large number of strategy configurations, in contrast to a single specification of (potentially adaptive) behavior found in other agent-based modeling methodologies. Consequently, EGTA can be more computationally expensive than other agent-based modeling methodologies. As an additional layer on top of agent-based simulation, it is also more complicated to conduct EGTA studies well, as many of the pitfalls and sources of error from ABM are still present, along with new concerns arising from comparing the outcomes of many simulations. These hurdles are worth overcoming, as EGTA enables the application of widely accepted tools from game theory to ABM, supporting strategic analysis with greater fidelity than is tractable for games defined analytically. Furthermore, improving the ability to scale EGTA increases the capability to build more sophisticated models, better approximating the strategic environments being studied. Through building an intelligent software system to support this methodology, incorporating best practices for sampling, statistics, and analysis, the learning curve and the cost of employing EGTA at scale can be improved. The benefit of this approach is difficult to measure, but can be seen in the variety and scope of EGTA studies conducted using EGTAOnline to date.

BIBLIOGRAPHY

Samuel Alberts, Michael K. Keenan, and Roshan M. D'Souza. Data-parallel techniques for simulating a mega-scale agent-based model of systemic inflammatory response syndrome on graphics processing units. *Simulation*, 88(8):895–907, 2012.

Luciano Bononi, Michele Bracuto, Gabriele D'Angelo, and Lorenzo Donatiello. Concurrent replication of parallel and distributed simulations. In *19th Workshop on Principles of Advanced and Distributed Simulation*, pages 234–243, Monterey, California, 2005.

Tilman Borgers, Ingemar Cox, Martin Pesendorfer, and Vaclav Petricek. Equilibrium bids in sponsored search auctions: Theory and evidence. *American Economic Journal: Microeconomics*, 5(4):163–187, 2013.

Ben-Alexander Cassell and Michael P. Wellman. Asset pricing under ambiguous information: An empirical game-theoretic analysis. *Computational and Mathematical Organization Theory*, 18(4):445–462, 2012.

Ben-Alexander Cassell and Michael P. Wellman. EGTAOnline: An experiment manager for simulation-based game studies. In *Multi-Agent-Based Simulation VIII*, volume 7838 of *Lecture Notes in Computer Science*, pages 85–100. Springer Berlin Heidelberg, 2013.

Ben-Alexander Cassell and Michael P. Wellman. Database modeling of empirical games. In *1st Workshop on Data Science for Macro-Modeling with Financial and Economic Datasets*, Snowbird, UT, 2014.

Ben-Alexander Cassell, Timur Alperovich, Michael P. Wellman, and Brian Noble. Access point selection under emerging wireless technologies. In *Sixth Workshop on the Economics of Networks, Systems, and Computation*, San Jose, California, 2011.

Matteo Cesana, Ilaria Malanchini, and Antonio Capone. Modelling network selection and resource allocation in wireless access networks with non-cooperative games. In *5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, pages 404–409, Atlanta, GA, 2008.

Ranveer Chandra, Paramvir Bahl, and Pradeep Bahl. MultiNet: Connecting to multiple IEEE 802.11 networks using a single wireless card. In *Twenty-Third Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 882–893, Hong Kong, 2004.

David A. Chapman and Valery Polkovnichenko. First-order risk aversion, heterogeneity, and asset market outcomes. *Journal of Finance*, 64(4):1863–1887, 2009.

Shih-Fen Cheng, Daniel M Reeves, Yegeniy Vorobeychik, and Michael P. Wellman. Notes on equilibria in symmetric games. In *AAMAS-04 Workshop on Game Theoretic and Decision Theoretic Agents*, 2004.

E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

David Colander, Michael Goldberg, Armin Haas, Katarina Juselius, Alan Kirman, Thomas Lux, and Brigitte Sloth. The financial crisis and the systemic failure of the economics profession. *Critical Review: A Journal of Politics and Society*, 21(2-3):249–267, 2009.

John Collins, Wolfgang Ketter, and Anuraag Pakanati. An experiment management framework for TAC SCM agent evaluation. In *IJCAI-09 Workshop on Trading Agent Design and Analysis*, pages 9–13, Pasadena, California, 2009.

Daniele Croce, Emilio Leonardi, and Marco Mellia. Large-scale available bandwidth measurements: Interference in current techniques. *IEEE Transactions on Network and Service Management*, 8(4):361–374, 2011.

Pranav Dandekar, Ashish Goel, Michael P. Wellman, and Bryce Wiedenbeck. Strategic formation of credit networks. In *21st International Conference on World Wide Web*, pages 559–568, Lyon, France, 2012.

Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papdimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009.

A. C. Davison and D. V. Hinkley. *Bootstrap Methods and their Application*. Cambridge University Press, 1997.

J. Bradford DeLong and Konstantin Magin. The U.S. equity return premium: Past, present, and future. *Journal of Economic Perspectives*, 23(1):193–208, 2009.

Larry G. Epstein and Martin Schneider. Ambiguity, information quality, and asset pricing. *Journal of Finance*, 63(1):197–228, 2008.

Jon Exley, Shyam Mehta, and Andrew Smith. Mean reversion. In *Finance and Investment Conference*, Brussels, Belgium, 2004.

Eugene F. Fama and Kenneth R. French. The equity premium. *Journal of Finance*, 57: 637–659, 2002.

Sevan Ficici, David C. Parkes, and Avi J. Pfeffer. Learning and solving many-player games through a cluster-based representation. In *Twenty-fourth Conference on Uncertainty in Artificial Intelligence*, pages 187–195, Helsinki, Finland, 2008.

Daniel Friedman. Evolutionary games in economics. *Econometrica*, 59(3):637–666, 1991.

Daniel Friedman and John Rust, editors. *The Double Auction Market: Institutions, Theories, and Evidence*. Addison-Wesley, 1993.

M. D. Gerst, P. Wang, A. Roventini, G. Fagiolo, G. Dosi, R. B. Howarth, and M. E. Borsuk. Agent-based modeling of climate policy: An introduction to the ENGAGE multi-level model framework. *Environmental Modelling & Software*, 44:62–75, 2013.

B. K. Ghosh and P. K. Sen, editors. *Handbook of Sequential Analysis*, chapter 1: A Brief History of Sequential Analysis. Marcel Dekker, 1991.

Itzhak Gilboa and David Schmeidler. Maxmin expected utility with non-unique prior. *Journal of Mathematical Economics*, 18(2):141–153, 1989.

Domenico Giustiniano, Eduard Goma, Alberto Lopez Toledo, and Pablo Rodriguez. WiSwitcher: An efficient client for managing multiple APs. In *Second ACM SIGCOMM Workshop on Programmable Routers for Extensible Services of Tomorrow*, pages 43–48, 2009.

Yoram Halevy. Ellsberg revisited: An experimental study. *Econometrica*, 75:503–536, 2007.

Joseph M. Hellerstein, Christopher Ré, Florian Schoppmann, Daisy Zhe Wang, Eugene Fratkin, Aleksander Gorajek, Kee Siong Ng, Caleb Welton, Xixuan Feng, Kun Li, and Arun Kumar. The MADlib analytics library: or MAD skills, the SQL. *Proceedings of the VLDB Endowment*, 5(12):1700–1711, August 2012.

Christoper Jennison and Bruce W. Turnbull. Repeated confidence intervals for group sequential clinical trials. *Controlled Clinical Trials*, 5(1):33–45, 1984.

Albert Xin Jiang, Kevin Leyton-Brown, and Navin A. R. Bhat. Action-graph games. *Games and Economic Behavior*, 71:141–173, 2011.

Patrick R. Jordan and Michael P. Wellman. Generalization risk minimization in empirical game models. In *Eighth International Conference on Autonomous Agents and Multi-Agent Systems*, pages 553–560, Budapest, Hungary, 2009.

Patrick R. Jordan, Christopher Kiekintveld, and Michael P. Wellman. Empirical game-theoretic analysis of the TAC supply chain game. In *Sixth International Joint Conference on Automomous Agents and Multi-Agent Systems*, pages 1188–1195, Honolulu, Hawaii, 2007.

Patrick R. Jordan, Yevgeniy Vorobeychik, and Michael P. Wellman. Searching for approximate equilibria in empirical games. In *Seventh International Conference on Autonomous Agents and Multi-Agent Systems*, pages 1063–1070, Estoril, Portugal, 2008.

Patrick R. Jordan, L. Julian Schvartzman, and Michael P. Wellman. Strategy exploration in empirical games. In *Ninth International Conference on Autonomous Agents and Multiagent Systems*, pages 1131–1138, Toronto, Canada, 2010.

Michael Kifer, Arthur Bernstein, and Philip M. Lewis. *Database Systems: an application-oriented approach*, chapter 11: An Overview of Query Optimization. Pearson Education, 2nd edition, 2005a.

Michael Kifer, Arthur Bernstein, and Philip M. Lewis. *Database Systems: an application-oriented approach*, chapter 9: Physical Data Organization and Indexing. Pearson Education, 2nd edition, 2005b.

Robert Kleinberg, Georgios Piliouras, and Eva Tardos. Load balancing without regret in the bulletin board model. In *Twenty-Eighth ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 56–62, Calgary, Canada, 2009.

Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. *Computer Science Review*, 3(2):65–69, 2009.

Elias Koutsoupias, Panagiota N. Panagopoulou, and Paul G. Spirakis. Selfish load balancing under partial knowledge. In *Mathematical Foundations of Computer Science*, volume 4708 of *Lecture Notes in Computer Science*, pages 609–620. Springer Berlin Heidelberg, 2007.

S. S. Lavenberg and P. D. Welch. A perspective on the use of control variables to increase the efficiency of Monte Carlo simulations. *Management Science*, 27(3):322–335, 1981.

Blake LeBaron. Agent-based computational finance. In Leigh Tesfatsion and Kenneth L. Judd, editors, *Handbook of Agent-Based Computational Economics*. Elsevier, 2006.

Pierre L'Ecuyer. Efficiency improvement and variance reduction. In *Twenty-Sixth Winter Simulation Conference*, pages 122–132, Orlando, FL, 1994.

Roberto Leombruni and Matteo Richiardi. Why are economists sceptical about agent-based simulations? *Physica A: Statistical Mechanics and its Applications*, 355(1):103–109, 2005.

Francis A. Longstaff. Asset pricing in markets with illiquid assets. In *American Finance Association 2006 Boston Meetings*, 2006.

Richard D. McKelvey, Andrew M. McLennan, and Theodore L. Turocy. Gambit: Software tools for game theory, version 14.0.1. http://www.gambit-project.org/.

Richard D. McKelvey, Andrew M. McLennan, and Theodore L. Turocy. Gambit: Software tools for game theory. Technical report, Version 0.2006.01.20, 2006. URL http://econweb.tamu.edu/gambit/.

Rajnish Mehra and Edward C. Prescott. The equity premium: A puzzle. *Journal of Monetary Economics*, 15(2):145–161, 1985.

Dawit Mengistu, Paul Davidsson, and Lars Lundberg. Middleware support for performance improvement of MABS applications in the grid environment. In *Multi-Agent-Based Simulation VIII*, volume 5003 of *Lecture Notes in Computer Science*, pages 20–35. Springer Berlin / Heidelberg, 2008.

Kimaya Mittal, Elizabeth M Belding, and Subhash Suri. A game-theoretic analysis of wireless access point selection by mobile users. *Computer Communications*, 31:2049–2062, 2008.

Michael Mitzenmacher. How useful is old information? In *16th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 83–91, Santa Barbara, 1997.

John Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.

A. J. Nicholson, Y. Chawathe, M. Y. Chen, B. D. Noble, and D. Wetherall. Improved access point selection. In *4th International Conference on Mobile Systems, Applications, and Services*, pages 233–245, Uppsala, Sweden, 2006.

Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007.

Eugene Nudelman, Jennifer Wortman, Yoav Shoham, and Kevin Leyton-Brown. Run the GAMUT: A comprehensive approach to evaluating game-theoretic algorithms. In *Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 880–887, New York, New York, 2004.

Marco Raberto, Andrea Teglio, and Silvano Cincotti. Integrating real and financial markets in an agent-based economic model: An application to monetary policy design. *Computational Economics*, 32:147–162, 2008.

Daniel M. Reeves, Michael P. Wellman, Jeffrey K. MacKie-Mason, and Anna Osepayshvili. Exploring bidding strategies for market-based scheduling. *Decision Support Systems*, 39(1):67–85, 2005.

Patrick F. Riley and George F. Riley. Spades: A distributed agent simulation environment with software-in-the-loop execution. In *35th Winter Simulation Conference*, pages 817–825, New Orleans, LA, 2003.

Michael Rothschild and Joseph E. Stiglitz. Increasing risk: I. A definition. *Journal of Economic Theory*, 2:225–243, 1970.

Peter Schuster and Karl Sigmund. Replicator dynamics. *Journal of Theoretical Biology*, 101:19–38, 1983.

Srinivas Shakkottai, Eitan Altman, and Anurag Kumar. Multihoming of users to access points in WLANs: A population game perspective. *IEEE Journal on Selected Areas in Communications*, 25:1207–1215, 2007.

Matthias Sheutz and Jack J. Harris. An overview of the SimWorld agent-based grid experimentation system. In *Large-Scale Computing Techniques for Complex System Simulations*. John Wiley & Sons, 2012.

Michael Stonebraker. SQL databases v. NoSQL databases. *Communications of the ACM*, 53(4):10–11, 2010.

Subhash Suri, Csaba D. Toth, and Yunhong Zhou. Selfish load balancing and atomic congestion games. In *Sixteenth ACM Symposium on Parallelism in Algorithms and Architectures*, pages 188–195, Barcelona, Spain, 2004.

Stefan Thurner. Systemic financial risk: Agent based models to understand the leverage cycle on national scales and its consequences. Technical Report IFP/WKP/FGS(2011)1, Organisation for Economic Co-operation and Development (OECD), 2011.

Yegeniy Vorobeychik and Michael P. Wellman. Strategic analysis with simulation-based games. In *41st Winter Simulation Conference*, pages 359–372, 2009.

Yevgeniy Vorobeychik, Michael P. Wellman, and Satinder Singh. Learning payoff functions in infinite games. *Machine Learning*, 67:145–168, 2007.

Elaine Wah and Michael P. Wellman. Welfare effects of market making in continuous double auctions (preliminary report). In *16th International Workshop on Agent-Mediated Electronic Commerce and Trading Agents Design and Analysis (AMEC/TADA)*, Paris, France, 2014.

Martin L. Weitzman. Subjective expectations and asset-return puzzles. *American Economic Review*, 97:1102–1130, 2007.

Michael P. Wellman. Methods for empirical game-theoretic analysis. In *21st National Conference on Artificial Intelligence*, pages 1552–1555, Boston, MA, 2006.

Michael P. Wellman and Bryce Wiedenbeck. An empirical game-theoretic analysis of credit network formation. In *Fiftieth Annual Allerton Conference on Communication, Control, and Computing*, Urbana, Il, 2012.

Michael P. Wellman, Daniel M. Reeves, Kevin M. Lochner, Shih-Fen Cheng, and Rahul Suri. Approximate strategic reasoning through hierarchical reduction of large symmetric games. In *Twentieth National Conference on Artificial Intelligence*, pages 502–508, Pittsburgh, PA, 2005.

Michael P. Wellman, Eric Sodomka, and Amy Greenwald. Self-confirming price prediction strategies for simultaneous one-shot auctions. In *28th Conference on Uncertainty in Artificial Intelligence*, pages 893–902, Catalina Island, CA, 2012.

Michael P. Wellman, Tae Hyung Kim, and Quang Duong. Analyzing incentives for protocol compliance in complex domains: A case study of introduction-based routing. In *12th Workshop on the Economics of Information Security*, Washington, D.C., 2013.

John Whitehead. On the bias of maximum likelihood estimation following a sequential test. *Biometrika*, 73(3):573–581, 1986.

Jonathan Widger and Daniel Grosu. Computing equilibria in bimatrix games by parallel support enumeration. In *Seventh International Symposium on Parallel and Distributed Computing*, pages 250–256, Krakow, Poland, 2008.

Bryce Wiedenbeck and Michael P. Wellman. Scaling simulation-based game analysis through deviation-preserving reduction. In *Eleventh International Conference on Autonomous Agents and Multiagent Systems*, pages 931–938, Valencia, Spain, 2012.

Bryce Wiedenbeck, Ben-Alexander Cassell, and Michael P. Wellman. Bootstrap statistics for empirical games. In *13th International Conference on Autonomous Agents and Multiagent Systems*, pages 597–604, Paris, France, 2014.

Fengyuan Xu, Chiu C Tan, Qun Li, Guanhua Yan, and Jie Wu. Designing a practical access point association protocol. In *29th IEEE Conference on Computer Communcations*, San Diego, CA, 2010.