EGTAOnline: An Experiment Manager for Simulation-Based Game Studies

Ben-Alexander Cassell and Michael P. Wellman

Computer Science & Engineering
University of Michigan
Ann Arbor, MI 48109-2121 USA

Abstract. Empirical game-theoretic analysis (EGTA) is a promising methodology for studying complex strategic scenarios through agent-based simulation. One challenge of utilizing this methodology is that it can require tremendous amounts of computation. Constructing the payoff matrix for a game of even moderate complexity entails significant data gathering and management concerns. We present EGTAOnline, an experiment management system that simplifies the application of the EGTA methodology to large games. We describe the architecture of EGTAOnline, explain why such a tool is practically important, and discuss avenues of research that are suggested through the use of EGTAOnline.

1 Introduction

We are living in a golden age for distributed computing. 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 efficient 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 computer. Such restrictions may detract from real-world relevance of computational investigations, or otherwise reduce the value of published studies.

One area where scope limitations can significantly weaken conclusions is in the analysis of *empirical games*: game-theoretic models induced from observations of multiagent interactions. Interaction data could be drawn from real-world

[1] `http://aws.amazon.com`

observations, but more commonly come from agent-based simulations. As for any game model, the empirical game maps *strategy profiles* (joint strategy configurations) of the agents (*players*, in game theory terminology) to payoff values representing the utility accrued by the respective agent for playing its strategy in that profile. The space of possible profiles grows exponentially with the number of players and the number of strategies, pushing the construction of games of even moderate complexity beyond the reach of a typical desktop computer, without a robust data storage and retrieval solution. Even if a researcher resolves the problem of storing and organizing observation data, the time required to generate observations for thousands or even millions of profiles on a single desktop computer, where each sample could take as long as an hour to gather, could outstrip the researcher's lifetime, making it impossible to meet any publication deadlines. As such, some mechanism for managing the distribution of game simulations and retrieving observations is needed.

We present EGTAOnline, an experiment management system designed to make studying large empirical games, derived from agent-based simulations, more convenient. Our current implementation of EGTAOnline strives to make the most common aspects of employing the empirical game-theoretic analysis (EGTA) methodology available through simple web forms, while supporting more complex functionality through a JSON[2] API. Following a review of related efforts, we present the EGTA methodology and detail how the EGTAOnline architecture supports the application of this methodology. We then discuss how to employ EGTAOnline to automate game refinement, leading to new areas of research, and describe the usage of our system to date.

2 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 simulation (MAS) over multiple computers. Lees et al. (2005) argued that conventional parallel discrete-event simulation techniques were lacking when it came to distributed MAS, and propose Communication Logical Processes to manage the shared state associated with MAS. Mengistu et al. (2008) identify several architectural issues in designing MAS for Grid computing, including threading and communication overhead, and present middleware to address some of these challenges. 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.

EGTAOnline builds on a tradition of tool-building in the computational game theory community. McKelvey et al. (2006) describe Gambit, a collection

[2] `http://www.json.org`

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 related systems, developed by Jordan et al. (2007) and Collins et al. (2009), that provided web interfaces for scheduling simulations of a supply chain management game. Our current system was constructed in part because scheduling and data management issues arose while conducting EGTA studies of the equity premium in financial markets (Cassell and Wellman, 2011) and wireless access point selection (Cassell et al., 2011). It became clear that an experiment manager with the capabilities described here would have greatly facilitated that work.

3 Empirical Game-Theoretic Analysis

Empirical game-theoretic analysis (EGTA)(Wellman, 2006) is an emerging methodology for examining complex strategic interactions between multiple agents. Fundamentally, EGTA applies the analytical tools of game theory to games that are constructed from empirical observations of strategic play. These observations may come from real-world data or from agent-based simulation. The EGTAOnline system supports development of empirical games induced from simulation.

Formally, a normal-form game Γ is specified by the tuple $[I, \{S_i\}, u(\cdot)]$. In this description, I is the set of players of the game and S_i is the set of strategies that player $i \in I$ may play. A profile of Γ, $s = \left(s_1, \ldots, s_{|I|}\right)$, assigns a strategy to each player. The function $u(\cdot)$ maps a profile of Γ to the payoff each player receives for playing their assigned strategy in the profile. This description implies an $|I|$-dimensional payoff matrix, where entries are payoff vectors in $\mathbb{R}^{|I|}$, indexed by the corresponding profile.

We can often achieve a more compact game model by exploiting symmetry in the set of players. A *role-symmetric game* is a tuple $\Gamma = [\{I_j\}, \{S_j\}, u(\cdot)]$, where the set of players with role j is I_j, and players with role j have strategy set S_j. Role-symmetric games provide a natural model for many settings where agents can be partitioned into meaningful categories, such as buyers and sellers in a market, or attackers and defenders in a security game. Assuming role symmetry is without loss of generality, as a game with no symmetry can be expressed by assigning each player its own role. At the other end of the spectrum, a fully symmetric game is one where all players have the same role. Between these two extremes are games with multiple roles, and multiple players in some or all roles. In a role-symmetric game, the payoff for playing a strategy for a role depends only on how many other players of each role play each strategy, and is invariant to which of the players within a given role play those strategies.

Vorobeychik and Wellman (2008) introduce a description for simulation-based games that replaces $u(\cdot)$ with $\mathcal{O}$, an oracle that returns sample payoff observations such that, for any profile s, $E\left[\mathcal{O}(s)\right] = u(s)$. 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 payoff function $u(\cdot)$ for the game we are simulating. The simulator may be noisy, necessitating repeated sampling and possibly the application of statistical techniques to ensure accurate payoff approximations.

Figure 1 outlines the basic procedure of applying EGTA to a simulation-based game. Given an appropriate simulator, we first propose a set of heuristic strategies that agents may play. This set of strategies induces a space of profiles. From this space of profiles, we select profiles to observe in the simulator. Given a profile as input, the simulator outputs the observed payoff each agent received for playing their specified strategy in the profile. This output is used to update the payoff estimates for that profile in the empirical game. At any point, we may choose to refine our empirical game by taking more samples of certain profiles, or increasing the strategy set and thus the profile space, possibly using game-theoretic analysis of the current empirical game to inform these decisions. Once we are finished refining the empirical game, we can report the findings of our game-theoretic analysis.

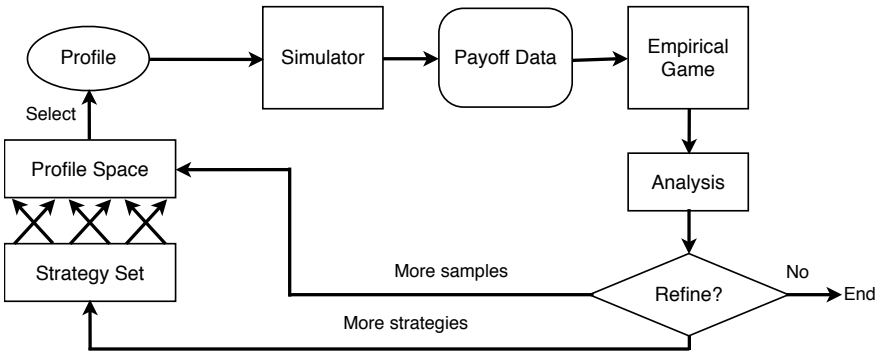

Fig. 1. Basic flow diagram of the iterative EGTA process.

4 EGTAOnline

EGTAOnline is an experiment manager for simulation-based game studies. It provides researchers with distributed simulation scheduling and a robust data storage solution. Users of EGTAOnline can take advantage of the parallel 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 query language. As such, barriers to constructing large simulation-based games are dramatically reduced.

EGTAOnline also supports a richer description of simulation-based games than what is suggested by the oracle-based game definition of Vorobeychik and

Wellman (2008). A normal-form game in EGTAOnline can be described by the tuple $[\{I_j\}, \{S_j\}, \mathcal{O}_c, P]$. Here, c specifies a configuration of run-time parameters of the simulator $\mathcal{O}$, and P is a collection of sample observations from $\mathcal{O}_c$, in the role-symmetric profile space defined by $\{I_j\}$ and $\{S_j\}$. This description explicitly maintains distributional information about the sample observations, and allows the possibility of partial games: games with profiles that have not yet been sampled. Additionally, we do not restrict sample observations to express only payoffs. They may carry additional information about the realizations of random variables, which can be useful in the application of statistical methods to the observation data.

Figure 2 illustrates the role of EGTAOnline in the iterative EGTA process. The following subsections present the primary conceptual entities of EGTAOnline and how they support the construction of empirical games.

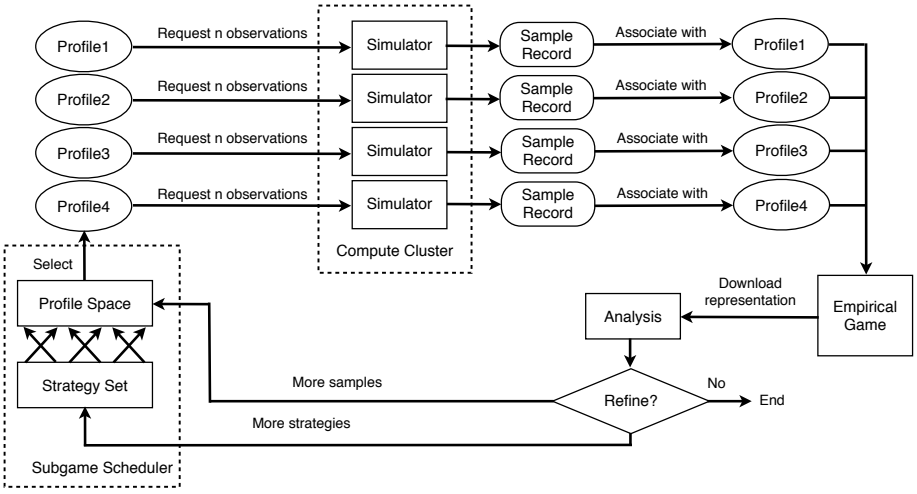

Fig. 2. Supporting the EGTA process with EGTAOnline.

4.1 Simulators

To use EGTAOnline, researchers must write a *simulator* that acts as an oracle, taking a profile to sample as input, and outputting sample payoff information for that profile. Observations of other features may be associated with payoff data to enable statistical procedures, such as the use of control variates (Lavenberg and Welch, 1981) for reducing variance in payoff estimates. 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. Optionally, researchers may write their simulators to expose additional

run-time parameters through the EGTAOnline interface, allowing multiple game environments to be represented using a single simulator.

4.2 Schedulers

Once a simulator has been registered with EGTAOnline, the experimenter may create one or more *schedulers* for that simulator. Schedulers are responsible for translating user-specified constraints into simulation jobs that are scheduled onto a cluster. Specifically, schedulers take as input

1. running requirements, such as memory and time, that the simulator needs to take a sample
2. sampling information, such as maximum number of samples to gather per profile, and number of samples to gather per job request.
3. c, the configuration of run-time parameters to use with the simulator
4. $\{s\}$, 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 scheduler*, generates profiles defining a subgame by specifying a partition of players into roles, and restricted strategy sets $S'_j \subseteq S_j$ for each role j. Subgame schedulers construct the profile space associated with $\{S'_j\}$ by generating the set of all symmetric assignments of strategies in S'_j to players in j, for each role j, and taking the cartesian product of these sets.

The second, *deviation scheduler*, expands the set of profiles generated by a subgame scheduler through the addition of deviating strategy sets. Users specify a partition of players into roles, and two disjoint, restricted strategy sets, S'_j and S''_j, for each role j. The set $\{S'_j\}$ is used to generate the same set of profiles as in the subgame scheduler, while S''_j specifies strategies to which a player in role j may deviate. To the subgame induced by $\{S'_j\}$ are added any profiles that can be reached through one player switching to a strategy in its deviating strategy set. This scheduler is useful for incrementally searching for payoff-improving strategy deviations without committing to constructing the exponentially larger subgame induced by adding these strategies to the base strategy sets.

The third, *hierarchical scheduler*, allows users to request all profiles that are consistent with a *hierarchical game reduction* (Wellman et al., 2005). The hierarchical approximation scheme defines a reduced game with fewer players, where each controls multiple agents in the simulator of the full game. The hierarchical scheduler generates all profiles of a game that correspond to a pattern with a reduced number of players each choosing a strategy for a multiple of players in the actual game. EGTAOnline also supports a hierarchical deviation scheduler that schedules single-player deviations in the hierarchically reduced game.

To enable arbitrary profile sampling behavior, EGTAOnline allows users to construct *generic schedulers*. Profiles to sample, and the number of samples requested, are passed to these schedulers through a JSON API. Users can write

scripts with complex logic determining which profiles to sample, then send an HTTP request to update the scheduler accordingly. This flexibility enables automated refinement of empirical games, discussed further in Section 5.

4.3 Simulations

Simulations in EGTAOnline provide summary information of simulation jobs that have been scheduled on the cluster. Simulations display the status of the job and provide easy access to any error messages. Errors can be caused by system problems, such as connectivity interruptions, simulator failures, or any user supplied 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, while too rare to show up in preliminary testing, manifest themselves when many samples must be gathered.

4.4 Profiles

An EGTAOnline *profile* object is unique up to its simulator, simulator configuration, and strategy assignment. Each profile is associated with a collection of *sample records*. A sample record stores a payoff observation for the associated profile, along with any feature data recorded.

Distinguishing profiles by simulator and configuration enables consistent maintenance of data from many experiments. Data gathered from one version of a simulator may not be comparable to data gathered with another, and thus are stored with separate sets of profiles. Similarly, different configurations of a simulator correspond to different experimental setups, and as such, require separate profile sets. Conversely, when a profile already exists for a given simulator, configuration, and strategy assignment, any new data gathered is associated with that profile. Thus, profiles can be in the sampling set of multiple schedulers, and associated with multiple games, allowing observational data to be included in all relevant analysis contexts.

4.5 Games

Games are dynamic entities in EGTAOnline, acting as filtered views onto the current data. All profiles matching a game's simulator and configuration, $\mathcal{O}_c$, are associated with that game in the database. When users request a representation of the game at time t, this set of profiles is filtered by the specifications of $\{I_j\}$ and $\{S_j\}$ in effect at t. These profiles carry with them all the associated observational data available at t. Hence, the empirical game that a particular game in EGTAOnline describes changes over time, as more data is gathered or different profile spaces are requested.

5 Automated Game Refinement

Typically, the game refinement step of the EGTA methodology requires human intervention. A researcher defines an experiment, performs the required simulation, and analyzes the resulting empirical game. At this point, the researcher either reports findings or sets up another experiment, repeating the previous steps. In theory, 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 researchers to construct complex scripts that interact with EGTAOnline through HTTP requests. We describe two applications of automated game refinement and how they would be implemented with EGTAOnline.

5.1 Exploration of Profile Space

A common application of game-theoretic analysis is identifying strategically stable profiles of a game, known as Nash equilibria. A Nash equilibrium profile is an assignment of strategies to players such that no player can benefit through unilateral deviation to a different strategy. Though the problem of finding all Nash equilibria of an arbitrary game requires having observations of every profile, finding and validating a single equilibrium can often be accomplished through observations of a smaller space.

Jordan et al. (2008) examine 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. (2008) is Minimum-Regret-First-Search (MRFS), which uses estimates of regret to guide search. The *regret* of a profile s, denoted $\epsilon(s)$, is the maximum improvement in payoff that a player can achieve through unilateral deviation. The key concept behind MRFS is that for every profile s, at any step in our search, we have a 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 set of profiles that can be reached through a single player deviating from 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 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 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 the queue do
    Select from queue the lowest ε̂(s) profile s not already in ℓ
    if s is confirmed then
      Remove s from queue and assign 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 the 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. The comparison of these variants in terms of steps required to find a Nash equilibrium is left for future work.

5.2 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 a simulation-based game are good approximations of equilibria of the game that the simulator is modeling? Since EGTAOnline maintains the full history of sample observations, we can pose the question of whether all deviations from a candidate equilibrium are statistically worse, to some level of significance. If the candidate equilibrium does not meet this significance criteria, sequential estimation techniques (Ghosh et al., 1997) could be used to choose how many additional samples to take. Figure 3 demonstrates how to conduct this sequential sampling procedure using EGTAOnline.

After gathering additional samples, payoff estimates for the game are updated, and thus equilibria candidates need to be recomputed. Vorobeychik (2010) demonstrates that regret in a simulation-based game almost surely converges to the regret in the underlying game as more samples are gathered, allaying concerns about the stability of the set of equilibria candidates. In other words, if an equilibrium candidate ceases to be a candidate after taking additional samples, then it is unlikely to be an equilibrium of the game that our simulator is modeling.

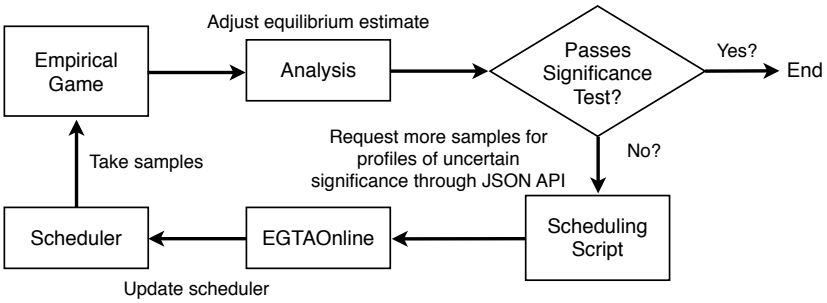

Fig. 3. Sequential sampling procedure to ensure statistical significance.

6 In Production

We developed EGTAOnline to address a perceived need for robust sampling infrastructure to support the EGTA methodology. One way to assess if we have achieved our goal is to observe how the system is used by practitioners of the methodology, and how heavily they use the system. Though we are currently exploring options for sharing EGTAOnline, to this point users have been limited to our lab and some direct collaborators. Over the last eight months of use, approximately 1.6 million sample records were recorded for 85,000 profiles. Many of these samples were generated for experiments detailed by Wellman et al. (2012) and Dandekar et al. (2012). Our database currently has eight distinct simulators registered, with multiple versions of some of these simulators. Schedulers (50, of which 29 are active) and games (76) significantly outnumber registered simulators (23), and have been used for exploring different simulator configurations

and different profile spaces. Users are free to modify or delete schedulers and games, making these numbers significant underestimates of the number of the experiments that have been carried out thus far. Though EGTAOnline in its current form has not been in use for very long, users are already taking advantage of its robust data storage system, and its support for the expanded definition of empirical games described in Section 3.

7 Discussion

Creating large empirical games through agent-based simulation carries many computational challenges. This does not mean, however, that we should not study large games. Many naturally occurring games, such as the stock market, are massive, and may be poorly modeled through analytical means or with small empirical games. Though we have the raw computation to begin modeling and analyzing the strategic implications of these massive social systems, the lack of convenient tools can make significant exploration a daunting task.

EGTAOnline is part of an ongoing effort to provide the necessary software infrastructure to make constructing and analyzing large simulation-based games more commonplace. Though new features are planned, we have demonstrated that EGTAOnline already supports the complex simulation and analysis workflows necessary for the application of the EGTA methodology. Indeed, we have presented a new way of specifying empirical games that allows for greater flexibility in terms of describing partially observed games, and describing multiple games derived from the same simulator, which is implemented in EGTAOnline. Additionally, our system opens new avenues of research through support for parallel profile sampling and automated empirical game refinement.

References

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

B.-A. Cassell and M. P. Wellman. Agent-based analysis of asset pricing under ambiguous information. In *SpringSim Agent-Directed Simulation Symposium*, pages 21–28, Boston, Massachusetts, 2011.

B.-A. Cassell, T. Alperovich, M. P. Wellman, and B. Noble. Access point selection under emerging wireless technologies. In *Sixth Workshop on the Economics of Networks, Systems, and Computation*, San Jose, California, 2011.

J. Collins, W. Ketter, and A. 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.

P. Dandekar, A. Goel, M. P. Wellman, and B. Wiedenbeck. Strategic formation of credit networks. In *21st International Conference on World Wide Web*, Lyon, France, 2012.

M. Ghosh, N. Mukhopadhyay, and P. K. Sen. *Sequential Estimation*. John Wiley & Sons, 1997.

P. R. Jordan, C. Kiekintveld, and M. P. Wellman. Empirical game-theoretic analysis of the TAC supply chain game. In *Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1188–1195, Honolulu, Hawaii, 2007.

P. R. Jordan, Y. Vorobeychik, and M. 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.

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.

M. Lees, B. Logan, R. Minson, T. Oguara, and G. Theodoropoulos. Distributed simulation of MAS. In *Multi-Agent and Multi-Agent-Based Simulation*, volume 3415 of *Lecture Notes in Computer Science*, pages 25–36. Springer Berlin / Heidelberg, 2005.

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

D. Mengistu, P. Davidsson, and L. 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.

E. Nudelman, J. Wortman, Y. Shoham, and K. 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.

M. Sheutz and J. 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.

Y. Vorobeychik. Probabilistic analysis of simulation-based games. *ACM Transactions on Modeling and Computer Simulation*, 20(3), September 2010.

Y. Vorobeychik and M. P. Wellman. Stochastic search methods for Nash equilibrium approximation in simulation-based games. In *Seventh International Conference on Autonomous Agents and Multi-Agent Systems*, pages 1055–1062, Estoril, Portugal, 2008.

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

M. P. Wellman, D. M. Reeves, K. M. Lochner, S.-F. Cheng, and R. Suri. Approximate strategic reasoning through hierarchical reduction of large symmetric games. In *20th National Conference on Artificial Intelligence*, pages 502–508, Pittsburgh, Pennsylvania, 2005.

M. P. Wellman, E. Sodomka, and A. Greenwald. Self-confirming price prediction strategies for simultaneous one-shot auctions. Technical report, University of Michigan, 2012.