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

From university-operated clusters to Amazon EC2 (`aws.amazon.com`), researchers increasingly have access to large pools of machines to aid in computational experimentation. Large-scale computing is increasingly available and inexpensive, yet human capital costs remain quite high. Researchers wishing to exploit available 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.

Scope limitations can significantly weaken conclusions in the analysis of *empirical games*: game-theoretic models induced from simulations of multiagent interactions. In contrast to other forms of agent-based modeling, empirical game-theoretic analysis (EGTA) addresses the question of strategy selection by comparing the payoffs obtained when agents play different configurations of strategies

in simulation. As for any game model, an 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 profiles grows exponentially with the numbers of players and strategies, pushing the construction of games of even moderate complexity beyond the reach of a typical desktop computer. Even if a single computer were fast enough to conduct the vast amounts of necessary simulation, the researcher may still experience data management concerns as storing all the observation data in memory will quickly impinge on the resources required to run further simulation, bringing the process of data acquisition to a grinding halt. As such, some mechanism for managing the distribution of game simulations and the retrieval of 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 methodology available through simple web forms, while supporting more complex functionality through a JSON (`www.json.org`) API.

Following a review of related efforts, we present the EGTA methodology and detail how the EGTAOnline architecture supports the application of this methodology. Afterwards, we discuss how EGTAOnline supports iterative experimentation through data reuse, and how the scheduling API enables automated game refinement, leading to new areas of research. Finally, we 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 based simulation (MABS) over multiple compute nodes. Riley and Riley (2003) presented 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) identified several architectural issues in designing MABS for grid computing, including threading and communication overhead, and presented middleware to address some of these challenges. Alberts et al. (2012) demonstrated that the parallelism afforded by modern graphics cards can be useful for simulations with millions of agents, as may be necessary when simulating 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 profile observations 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) described 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 game models from simulation data. EGTAOnline was also inspired by two existing 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 motivated in part by scheduling and data management issues that arose while conducting EGTA studies of the equity premium in financial markets (Cassell and Wellman, 2012) 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 (Wellman, 2006) 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 game models induced from simulation.

Formally, a normal-form game Γ is specified by the tuple $\langle I, \{S_i\}, u(\cdot)\rangle$. 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, $s = \big(s_1, \ldots, s_{|I|}\big)$, assigns a strategy to each player. Players may adopt a *mixed strategy*, a probability distribution over playing each of the strategies in their strategy set. When one or more players adopt a mixed-strategy, the joint-strategy selection is referred to as a *mixed-strategy profile*, and is otherwise referred to as a *pure-strategy profile*. The function $u(\cdot)$ maps a pure-strategy profile of Γ to the payoff each player receives for playing its assigned strategy in the profile, $\big(u_1(s), \ldots, u_{|I|}(s)\big)$. This description implies an $|I|$-dimensional payoff matrix, where entries are payoff vectors in $\mathbb{R}^{|I|}$, indexed by the corresponding profile. Player utilities for a mixed-strategy profile are calculated by taking the expectation of payoffs achieved under the pure-strategy profiles that can be realized under that mixed-strategy 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 $\Gamma = \langle \{I_j\}, \{S_j\}, u(\cdot)\rangle$, where the set of players are partitioned into roles such that players in role j all have the strategy set S_j. Role symmetry also constrains $u(\cdot)$ such that if two players in a role swap strategies, then their entries in the payoff vector are swapped and all other payoff entries are unaffected. As such, a player's payoff may depend only on its strategy choice, role membership, and how many players of each role play each strategy, being invariant to which of the players within a

given role play those strategies. 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.

Game models are used to predict agent behavior through the specification of a *solution concept*, a rule for calculating the probability that each pure-strategy profile will be played. 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 new strategy.

Vorobeychik and Wellman (2008) describe simulation-based games in terms of an oracle $\mathcal{O}$ that returns 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 payoff function $u(\cdot)$ for the game being simulated. The simulator may be noisy, necessitating repeated sampling to achieve accurate payoff estimates.

Figure 1 illustrates the basic EGTA procedure. Sets of heuristic strategies—one for each role—induce a space of profiles. We feed selected pure-strategy profiles to the game simulator, which outputs the observed payoff each agent received for playing its specified strategy in the profile. This output is used to update the payoff estimates for that profile in the empirical game model. At any point, we may choose to refine our game model by taking more samples of certain profiles, or adding more strategies and thus expanding the profile space, possibly using game-theoretic analysis of the current game model to inform these decisions. Once we have finished refining the empirical game model, we report the findings of our game-theoretic analysis.

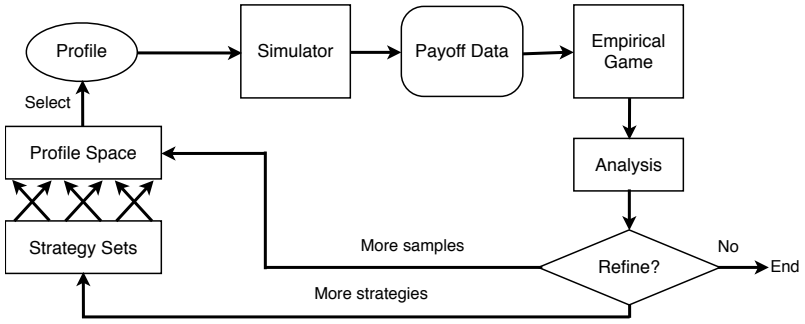

Fig. 1. Basic flow diagram of the iterative EGTA process.

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 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 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.

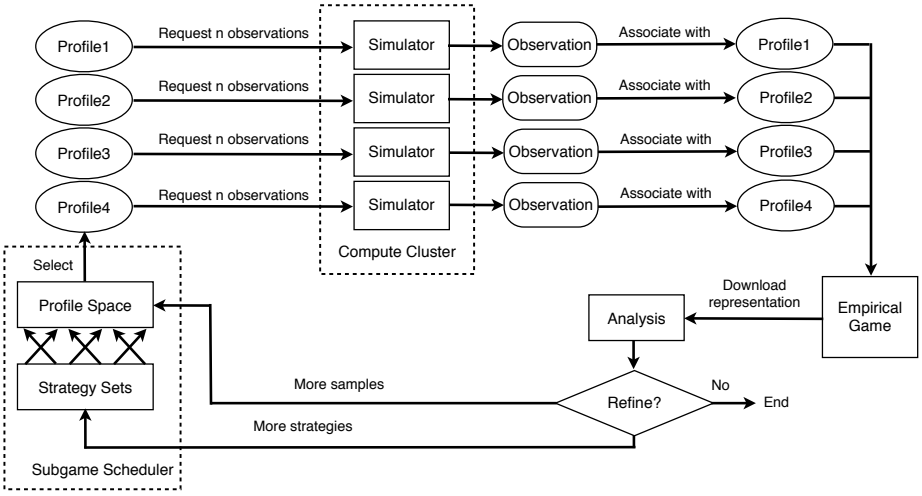

Fig. 2. Supporting the EGTA process with EGTAOnline.

4.1 Simulators

To use EGTAOnline, researchers must write a *simulator program* that acts as an oracle, taking as input a pure-strategy profile s to sample, and outputting an *observation*. An observation must include a vector of payoffs and may include statistics about the simulation. This statistical information can be useful in statistical procedures, such as the use of control variates (Lavenberg and Welch, 1981) for reducing variance in payoff estimates. These statistics may also record variables that resulted from agent interaction, enabling analysis of the impact of strategic choices on non-payoff variables. 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. A simulator program differs from the mathematical concept of a simulator described by Vorobeychik and Wellman (2008), as multiple simulators can be derived from a single simulator program through the specification of run-time parameters, or through specifying different assignments of players to roles. As such, a researcher can upload a single simulator program and perform multiple independent experiments.

4.2 Schedulers

Once a simulator program has been registered with EGTAOnline, the experimenter may create one or more *schedulers* for that simulator program. Schedulers translate user-specified requirements of the sampling process 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 an observation,
2. sampling information, such as maximum number of observations to gather per profile, and number of observations to gather per job request.
3. c, a configuration of run-time parameters to use with the simulator program, and
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*, generates profiles defining a subgame by specifying a partition of players into roles $\{I_j\}$, and restricted strategy sets $S'_j \subseteq S_j$ for each role j. Subgame schedulers construct the profile space associated with $\{I_j\}$ and $\{S'_j\}$ by generating the set of all symmetric assignments of strategies in S_j to players in I_j, for each role j, and taking the cartesian product of these sets.

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 j, two disjoint, restricted strategy sets: the base set S'_j and deviation set S''_j. The deviation scheduler uses the base sets $\{S'_j\}$ to generate profiles defining the full subgame over these strategies, as described above. To the subgame induced by $\{I_j\}$ and $\{S'_j\}$ are added any profiles that can be reached through one player switching to a strategy in its role's deviating strategy set, S''_j. This 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.

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. These approximations require exponentially fewer profiles than the corresponding full-player version of the game. EGTAOnline supports two types of reduction. In the *hierarchical* reduction (Wellman et al., 2005), each player controls the strategy played by multiple agents in simulation of a corresponding full game profile. For example, we may approximate an 8-player symmetric game with a 4-player game in which each player specifies the strategy to be played by two agents in simulation. In the *deviation-preserving* reduction (Wiedenbeck and Wellman, 2012), each player controls one agent, but models the remaining players as proportionally controlling the remaining agents. This reduction is so named because it emphasizes preserving single-player deviation incentives—precisely those incentives which are important for establishing a Nash equilibrium—while still aggregating payoffs over many agents. 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 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 reduction scheduler may also be used in larger reduced game models, as well as in unreduced game models.

When constructing either reduction for games with multiple roles, each role may reduce the number of players at different rates. 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 home buying 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.

To enable arbitrary profile sampling behavior, EGTAOnline allows users to specify *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 feature, combined with the ability to request game representations through the JSON API, provides the flexibility necessary to support automated refinement of game models, discussed further in Section 6.

4.3 Simulations

A *simulation object* in EGTAOnline summarizes the state of a simulation job that has been scheduled on the cluster. A simulation job can request multiple observations be taken for a single profile, amortizing the overhead of scheduling on the cluster. Simulation objects record the current status of a job and any associated error messages. 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, while too rare to show up in preliminary testing, manifest themselves when many observations must be gathered.

4.4 Profiles

An EGTAOnline *profile object* associates a collection of *observation objects* with the pure-strategy profile, simulator program, and configuration of run-time parameters that generated those observations. An observation object stores payoff and feature data for each player in the associated profile, as well as any other statistics, that were recorded during a single run of the simulator. Distinguishing profile objects by simulator program and configuration enables consistent maintenance of observation data from many experiments. Different configurations of a simulator program correspond to different experimental setups, and as such, require separate profile object sets. Conversely, when a profile object already exists for a given simulator program, configuration, and strategy assignment, any new data is associated with that profile object. 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—a topic we revisit in Section 5.

4.5 Games

A *game object* provides filtered views onto the current data. A game object defines a space of relevant profiles, based on a simulator program, configuration, 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 sent to the user in one of three available levels of detail. These profile objects carry with them all the associated observational data currently available, or summary statistics of the same.

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 game models. As such, we would like to maximize the value of the previously gathered data through extensive reuse. Data reuse is a natural consequence of the iterative EGTA process (Figures 1 and 2), 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 game-theoretic contexts. EGTAOnline accomplishes this by defining what it means to be *comparable*—sharing the same partition of players into roles, simulator program, and configuration. Consider two game objects differing only in their strategy sets. By the definition provided above, the observations associated with these two game objects are comparable. Thus, if both game objects specify a single role with a strategy set that includes 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, it will contain 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. Since most game analysis is super-linear in the number of profiles, game objects that restrict the strategy sets to only those currently under consideration take less time to assemble, download, and analyze.

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 determine whether such deviations refute 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.

In previous work, we presented two procedures where such data reuse can be extremely valuable (Cassell and Wellman, 2012). We described an equilibrium search technique that iterates through increasingly fine-grained game reductions, establishing a restricted set of promising strategies in smaller game abstractions to reduce the space of profiles needed to identify equilibria in more fine-grained abstraction. Since reduced-game profiles are represented in EGTAOnline in terms of their unreduced game constituents, each iteration of this search benefits from the data gathered in previous steps. If we are applying this technique to a symmetric game with N players and are contemplating sampling the profiles of an n-player hierarchical reduction of this game, where n divides N, then for each positive integer n' that divides n, any profile in the n'-player hierarchical reduction corresponds to an unreduced game profile that has a counterpart in the n-player reduction, and thus observations of such profiles can be reused. As such, if all smaller reductions have already been explored, the number of previously unobserved profiles of the n-player reduction is given by the recursive

relation

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

where m is the number of strategies.[1]

Figure 3 demonstrates the fraction of profiles that are covered by smaller reductions, $\sum_{n'|n} f(n',m)/\binom{n+m-1}{m-1}$, 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. Consequently, when n is prime, this value decreases as n increases, whereas the relationship is non-monotonic for composite values of n. For small values of m, the prospect of performing another iteration of the equilibrium search with a more fine-grained reduction is much less daunting, since 40–60% of the space may have been explored in earlier steps. Furthermore, this level of data reuse between steps increases the likelihood that the equilibrium candidates identified in each step are close to those identified in previous steps.

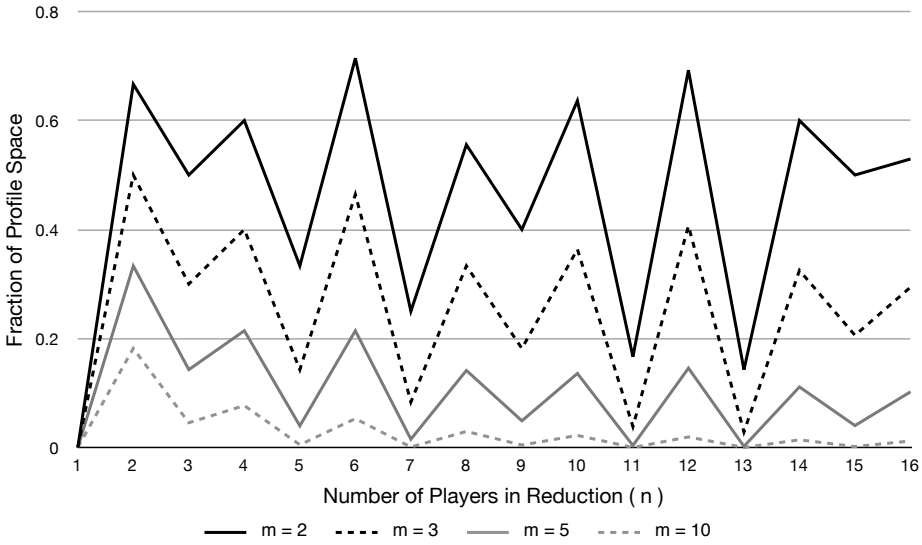

Fig. 3. Fraction of unreduced-game profile space for an n-player reduction that is covered by smaller reductions.

A second task that benefits from convenient data reuse is the estimation of expectations of non-payoff variables. In this context, reuse takes the form of deriving multiple interpretations of data from a single empirical game model. We may wish to know, for example, which of several possible configurations of

[1] Similar data reuse relationships can be constructed for role-symmetric hierarchical reductions and symmetric deviation preserving reductions.

strategies lead to the highest expected price volatility in financial markets. When we restrict our attention to pure-strategy profiles, this task may be as simple as comparing the sample averages of the variable of interest for each profile under consideration. For mixed-strategy profiles, estimating the expectation of a non-payoff variable requires weighting our observations by the probability that the associated pure-strategy profile is realized under the specified mixed-strategy profile. For a game Γ and a distribution σ specifying the probability that each pure-strategy profile is played, we can estimate the conditional expectation of a non-payoff variable V under σ by

$$\mathbb{E}\left[V \mid \sigma\right] = \sum_{s \in \Gamma} \bar{V}_s \sigma(s),$$

where $\bar{V}_s$ is the sample average of V when simulating the pure-strategy profile s and $\sigma(s)$ is the probability that s is realized under σ. By storing the observations of V with the profile that was played during the observation, we can calculate and compare expectation estimates of V under different distributions of pure-strategy profile realizations without additional simulation. As with the addition of a new strategy, if a new solution concept is proposed, and thus different distributions over pure-strategy profiles are predicted, we can use all of our previously gathered observations in constructing the new estimate.

6 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 model. 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.

6.1 Exploration of Profile Space

A common application of game-theoretic analysis is identifying Nash equilibria of a game. 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) 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 *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.

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.

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? Since EGTAOnline maintains the full history of observations and sampling decisions, we can pose the question

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 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
```

of whether all deviations from a candidate equilibrium are statistically worse, to some level of significance. Posing the problem this way, validating a Nash equilibrium is a form of simulation optimization (Ólafsson and Kim, 2002), which seeks to identify the best of several competing designs of a system or product through simulation. As such, we may appeal to the literature on optimal computing budget allocation (Chen and Lee, 2011), or the broader range of sequential estimation techniques (Ghosh et al., 1997), to decide how many additional samples of each profile to request at any decision point, as a function of statistical and strategic analysis of our accumulated game data. Figure 4 demonstrates how to conduct this sequential sampling procedure using EGTAOnline.

Confirming approximate Nash equilibria carries additional challenges not faced by more conventional sequential analysis problems. 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 observations 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 observations, then it is unlikely to be an equilibrium of the game that our simulator is modeling. Another challenge of using sequential techniques for this task is that our metric of interest, empirical regret, is unlikely to be well approximated by a simple distribution, constraining us to complicated, non-parametric procedures. To the best of our knowledge, the construction of optimal sequential sampling procedures for EGTA remains an open question.

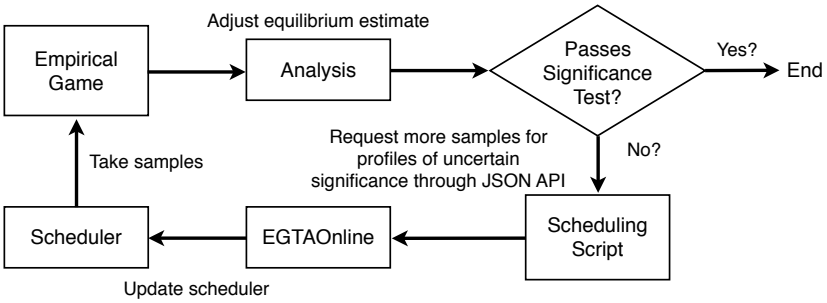

Fig. 4. Sequential sampling procedure to ensure statistical significance.

7 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 seventeen months of use, approximately 8 million observations were recorded for $300,000$ profiles. Many of these observations were generated for experiments detailed by Cassell and Wellman (2012), Wellman et al. (2012), and Dandekar et al. (2012). Our database currently has eleven distinct simulator programs registered, with multiple versions of some of these simulators. Schedulers (108) and games (95) significantly outnumber registered simulator programs (26), 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. There is considerable variety among the experiments conducted so far, ranging from explorations of the TAC Supply Chain Management game, where the simulation requires the parallel cooperation of multiple compute nodes, to an introduction-based routing protocol (Frazier et al., 2011), where role symmetry and hierarchical reduction are exploited. 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 high throughput.

8 Discussion

Creating large empirical game models 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 game models. 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. EGTAOnline also makes substantial data reuse practical, limiting duplication of effort and supporting iterative approaches to game exploration and experimentation. Additionally, our system opens new avenues of research through support for parallel profile sampling and automated game refinement, setting the stage for the development of intelligent agents that manage the iterative process of scheduling profiles to be sampled and analyzing the results.

References

S. Alberts, M. K. Keenan, and R. 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.

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, CA, 2005.

B.-A. Cassell and M. P. Wellman. Asset pricing under ambiguous information: An empirical game-theoretic analysis. *Computational and Mathematical Organization Theory*, 18:445–462, 2012.

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, CA, 2011.

C.-H. Chen and L. H. Lee. *Stochastic Simulation Optimization: An Optimal Computing Budget Allocation.* World Scientific Publishing Co., Singapore, 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.

G. Frazier, Q. Duong, M. P. Wellman, and E. Petersen. Incentivizing responsible networking via introduction-based routing. In *Fourth International Conference on Trust and Trustworthy Computing*, pages 277–293, Pittsburgh, 2011.

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, 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 Multiagent 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.

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, 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.

S. Ólafsson and J. Kim. Simulation optimization. In *Winter Simulation Conference*, pages 79–84, San Diego, 2002.

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

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), 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 Multiagent 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, 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, 2005.

M. P. Wellman, E. Sodomka, and A. Greenwald. Self-confirming price prediction strategies for simultaneous one-shot auctions. In *28th Conference on Uncertainty in Artificial Intelligence*, Catalina Island, CA, 2012.

B. Wiedenbeck and M. P. Wellman. Scaling simulation-based game analysis through deviation-preserving reduction. In *11th International Conference on Autonomous Agents and Multiagent Systems*, pages 931–938, Valencia, 2012.