EGTA

Elaine Wah
ewah@umich.edu

Updated: November 24, 2014

1 Introduction

The methodology of *empirical game theoretic analysis* (EGTA) [2] facilitates strategy selection for traders by comparing the payoffs (total surplus) of different combinations of traders and strategy assignments. EGTA is an iterative process to guide exploration of the strategy space, in which the goal is to confirm or refute equilibrium candidates.

We focus on *role-symmetric games*, in which agents' strategy sets are symmetric within a role, but may differ across roles. Payoffs in a role-symmetric game depends solely on role membership, strategy, and the number of agents playing the strategy. This permits us to exploit symmetry in certain types of agents in order to achieve a more compact game model.

To conduct and manage experiments, we use the EGTAOnline simulation manager, which allows us to generate simulation data for a variety of strategy profiles [1]. The testbed runs on the CAEN cluster flux. You can request access to flux by filling out this form. The process of uploading a version of the simulation system and scheduling simulations is described in Section 2. Game analysis is described in Section 3.

2 EGTAOnline

For more advanced users, many of the steps described below can be set up programmatically through the testbed's HTTP API.

2.1 Ready the simulator

To create and upload the simulator to the testbed, make sure you are in the `hft-sim` directory, then execute the following command:

```
./create-egta-simulator.sh <simulator name>
```

For example,

```
./create-egta-simulator.sh market_maker_sim
```

will create a zipped folder `market_maker_sim.zip`. The zipped simulator folder will include the basic configuration (i.e., the `env.properties` file) specified in the `egta` directory as well as the necessary `batch` script for running simulations on the cluster.

Note that you do not explicitly specify the path for the `defaults.json` file. Instead, the script will use `config/<simulator_name>.json`. In the example above, `config/market_maker_sim.json` will be copied into the zipped folder as `defaults.json`. This `json` file stores the simulator's default configurations, which will be the default values filled in whenever a new scheduler is created using your simulator. Format-wise, this `json` file looks exactly like the `simulation_spec.json` but with the "assignment" section removed.

> **Logging simulator versions:** Since each simulator's `defaults.json` file can vary widely depending on what environment parameters are needed, be sure to commit and push your updated simulator configuration files, all stored within the `config/` directory. In addition, whenever you upload a new simulator or update an existing simulator, *tag* the commit in the git repository. Tag format is as follows:
>
> `<simulator_name>-<version><upload_label>`
>
> For example, `market_maker_sim-2.4a` is the first tagged commit for version 2.4 of the `market_maker_sim` simulator. Any subsequent updates to this simulator will have incremented labels, e.g. `market_maker_sim-2.4b`. See the best practices documentation for details on tagging commits.

2.2 Upload your simulator

To create a new simulator on EGTAOnline, the following fields must be completed:

`Name:` Simulator name, e.g. `market_maker_sim`

`Version:` Simulator version, e.g. `2.0`

`Email:` Your email

`Zipped Source:` Select the zipped folder you have just created. The name of the `.zip` file should match the simulator name, i.e. you should be selecting `market_maker_sim.zip`

While it is possible to upload a new zipped folder to replace an existing simulator without changing the name or version number, it is of utmost importance than in such circumstances you do not change the strategies in a way that would invalidate data already collected.

After uploading the simulator, add the roles and corresponding strategies. These specify what roles and strategies will be available in the schedulers based on your simulator.

2.3 Create a scheduler

Schedulers are responsible for determining the profiles you need, given available strategies specified for each role, and acquiring the number of observations requested. A *profile* is an assignment of strategies to each player in a game.[1]

[1]For example, in the game of rock-paper-scissors, one profile may be represented as (R, P); this specifies the profile in which the first player plays 'Rock' and the second plays 'Paper'

As it is computational infeasible to analyze games with many traders, given the exponential growth in game size with the number of players, we employ *deviation-preserving reduction* or DPR [3]. Generally, player reduction is an aggregation technique that enables the approximation of games with many players via smaller games.

Other types of player reduction exist but for most purposes, you will be creating DPR schedulers or DPR deviation schedulers. Setting up the configurations is the same; the two types of schedulers differ primarily in how strategies are set up. To create a new DPR scheduler, the following fields need to be completed:

`Name:` Scheduler name (you will have many schedulers, so a naming system that includes short descriptors for the simulator, environment, and some numbering is recommended)

`Size:` Total number of agents (across all roles) in this scheduler; this is the number of players in the *full* game, not the reduced game

`Default observation requirement:` Number of observations the scheduler will obtain for each profile; increasing the `numSims` environment parameter and reducing this is recommended (choose values such that you obtain at least 200 runs per profile)

`Observations per simulation:` Given `numSims` ≥ 100, the recommended value for this parameter is at least 10

`Process memory:` Recommended value is 4000

`Time per observation (in seconds):` Depends on the strategies you plan to include, but note that the cluster will oftentimes be much slower than your own machine; for the purposes of the market maker simulations, for example, 400 to 500 is sufficient

`Nodes required:` For the HFT simulation system, 1 is sufficient

`Active:` Check box to activate the scheduler (and start running simulations). Leave this box unchecked for now. Do not activate the scheduler until you have set up all strategies and roles.

`Simulator:` Drop-down to select your simulator and version

The fields following `Simulator` will automatically update with the fields specified in `defaults.json`. If using non-default values, be very careful to avoid typos; otherwise, your simulation data will not be stored correctly in the database. After you set up the scheduler's strategies (see below), edit the simulator and check the `Active` box to start scheduling simulations.

DPR scheduler After creating a DPR scheduler, you will need to specify the roles, strategies, and number of agents. Select the role from the drop-down menu. The next two text boxes on the right allow you to specify the number of players in this role in the full game ("Count"), then the number of players in this role in the reduced game ("Reduced Count").

Once you've added a role, you can select the strategies to include in this scheduler. The scheduler will determine what profiles are needed based on the strategies selected for each role. The number of profiles needed grows exponentially with the number of strategies, so a general rule for the simulation system is to avoid including more than 3 trading strategies in any role with more than 1 player.

DPR deviation scheduler The purpose of the DPR deviation scheduler is to sample profiles that are single-player deviations from a candidate equilibrium. The *Roles and Strategies* section is set up similar to the DPR scheduler as described above, but be sure to specify only strategies from a candidate equilibrium. For example, if a candidate equilibrium (as determined from the analysis script) involves all background traders playing `ZIR:Rmin_0_Rmax_1000` and the market maker playing `MAMM:K_200`, pick only those strategies in this section, even if other strategies are present in the maximal subgame found. The *Deviating Roles and Strategies* section indicates the deviating strategies you would like to sample. For most purposes, you will need to include *every* available strategy (for each role) in this list, even if those strategies are not present in the maximal subgame.

2.4 Monitoring simulations

You can monitor your individual schedulers to see how many observations are still needed. The bottom of each scheduler's page will have a section called *Profiles* which indicates the strategy profiles that are combinations of the allowed strategy sets for each role. The "Requested Observations" column will match the `Default observation requirement`, and "Observation Count" indicates the number of observations you've collected so far for a given profile. You can click on the column header to sort.

Monitoring your experiments from the Simulations page is also necessary to ensure that your simulations are running as expected. There are a number of possible states you might see; if you see "failed" for any of your profiles, click the state to determine what the error message is. If you see an error message that mentions "Java," this indicates that there is most likely an error within the simulation system.

2.5 Constructing a game

You can construct a game directly from a scheduler's page (click "Create Game to Match"). This will take you directly to the game's page, from which you can add the strategies you'd like to include in this game. Multiple games can be constructed from the same data. For example, to examine the welfare effects of market making, we can construct two games—both with BACKGROUND and MARKETMAKER roles—in which the game without market making included only one strategy for market makers (the no-trade strategy NOOP), while the game with market making included all non-NOOP market maker strategies.

3 Game Analysis

In EGTA, a profile is confirmed when all possible deviations have been examined, and no beneficial deviation exists. As the observed payoffs from our simulator for a given profile are incrementally added, we analyze each successive intermediate game model and continue to refine the empirical game with the estimated payoffs until all candidate Nash equilibria are refuted or confirmed. The steps in this iterative process are described in Section 3.2.

We look at *regret* in the analysis script output to determine when a candidate equilibrium has been confirmed. Regret of a profile is the maximum amount of additional payoff that any player can gain from switching to a different strategy.

3.1 Analyzing games

There are two options for analyzing your game: directly through the testbed or manually on the command line.

3.1.1 Analysis on EGTAOnline

The easiest way to analyze your game is via the testbed. At the bottom of each game's page, click "Analyze JSON in EGAT" to go to the analysis setup page. Most default values will suffice, but you may wish to change some fields depending on your game:

`Scripts running time:` Sets the maximum running time for analysis; some games may take several hours to analyze, as clique-finding can be a bottleneck

`Enable reduction script:` Leave checked to analyze the reduced game

`Enable verbose flag:` Uncheck this box to make analysis output easier to read

`Regret:` Recommended initial max value of 0.1; increase if finding 0 approximate equilibria

`Points:` Increase if finding 0 approximate equilibria

3.1.2 Analysis on command line

The second option is to analyze your game via command line. The game analysis scripts are maintained by Bryce Wiedenbeck. To clone the repository for game analysis, navigate to the directory which holds the simulation system repository (e.g., `hft_project`), then execute the following command:

```
git clone https://github.com/egtaonline/GameAnalysis.git
```

This should clone the GameAnalysis repository at the same level as the hft repo. Be sure to pull periodically to ensure that the game analysis code is up to date.

To reduce the game and then analyze it:

1. First create a directory `games` in `hft_project`:

   ```
   mkdir games
   ```

2. Download your relevant game file: Click "Download JSON" from the specific game's page and move the resulting file (file name format will be `#-summary.json`) to a new folder in `games`. For example, if I download the summary of game 178 from the Games page, I move the resulting `178-summary.json` file from my default `Downloads` directory to `games/mmgame`.

3. Reduce the game to N_1, ..., N_r, where N_i indicates the number of players in the i-th role (when roles are sorted in alphabetical order). From the `hft_project` directory, use the following command to generate the reduced game:

   ```
   ./GameAnalysis/Reductions.py -input <summary game file>
       -output <reduced game file> DPR <N_1> <N_2> ... <N_r>
   ```

For example, if I have two roles, BACKGROUND and MARKETMAKER, and I would like to generate a reduced game with 6 of the former and 1 of the latter, I execute the following command:

```
./GameAnalysis/Reductions.py -input games/mmgame/178-summary.json -output
    games/mmgame/178-reduced.json DPR 6 1
```

4. Copy the game analysis script from the `GameAnalysis/scripts` directory to the `GameAnalysis` directory with the command:

   ```
   cp GameAnalysis/scripts/AnalysisScript.py GameAnalysis/
   ```

   The game analysis script can then be executed as:

   ```
   ./GameAnalysis/AnalysisScript.py [arguments] [reduced game file]
   ```

   For example,

   ```
   ./GameAnalysis/AnalysisScript.py -r 1 games/mmgame/178-reduced-json
   ```

   Add `> [output file name]` to the end of the command to pipe the output into a separate text file, such as:

   ```
   ./GameAnalysis/AnalysisScript.py -r 1 games/mmgame/178-reduced-json
       > games/mmgame/analysis_output.txt
   ```

3.2 EGTA inner loop

When analyzing your game, if 0 approximate mixed strategy Nash equilibria are found, try ramping up regret (e.g., use arguments `-r 1` or `-r 1000`) first, or increasing the number of starting points for replicator dynamics. Another option is to change the convergence threshold to something smaller (e.g., `-c 1e-10` or `-c 1e-12`). You may have to try a variety of parameters before any candidate equilibria are found.

More profiles are needed whenever you observe either of the following scenarios within the output of the analysis script:

`Deviation subgame UNEXPLORED!:` This indicates that a subgame needs to be explored. For example, given the following output for a subgame:

```
BACKGROUND:
    ZIR:Rmin_0_Rmax_5000: 100.0%
MARKETMAKER:
    NOOP: 100.0%
best responses:
        BACKGROUND: ZIR:Rmin_0_Rmax_1000;   gain = 16432.0212
Deviation subgame UNEXPLORED!
```

you will need to set up a DPR scheduler with two BACKGROUND strategies, `ZIR:Rmin_0_Rmax_5000` and `ZIR:Rmin_0_Rmax_1000`, as well as the MARKETMAKER strategy `NOOP`.

`regret >=` ϵ**:** A candidate equilibrium still needs to be confirmed or refuted it is lower bounded by some small ϵ close to 0, instead of `regret = 0.0` or `regret =` ϵ. In such cases, you will need to set up a DPR deviation scheduler with the candidate equilibrium in the *Roles and Strategies* section and with all other strategies available in that game in the *Deviating Roles and Strategies* section. For instance, `regret >= 0.0023` means the candidate equilibrium is unconfirmed, but `regret = 0.0023` means that it has been confirmed (even though it's not exactly 0).

You can verify to some degree that your new scheduler has been set up correctly by sorting by the number of observations ("Observation Count"). You should see at least one profile with 0 observations (indicating that you need to collect data for this profile).

After you've set up any new schedulers as needed and have collected data for additional profiles, repeat this entire procedure (starting with downloading the game) and repeat until the game has reached quiescence and all equilibrium candidates have been either confirmed or refuted.

References

[1] Ben-Alexander Cassell and Michael P. Wellman. EGTAOnline: An experiment manager for simulation-based game studies. In *Multi-Agent-Based Simulation XIII*, volume 7838. Springer, 2013.

[2] Michael P. Wellman. Methods for empirical game-theoretic analysis (extended abstract). In *21st National Conference on Artificial Intelligence*, pages 1552–1555, 2006.

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