AD Exchange Agent
Project Report

Chen Doar, Roei Mischori, Nizan Kramf

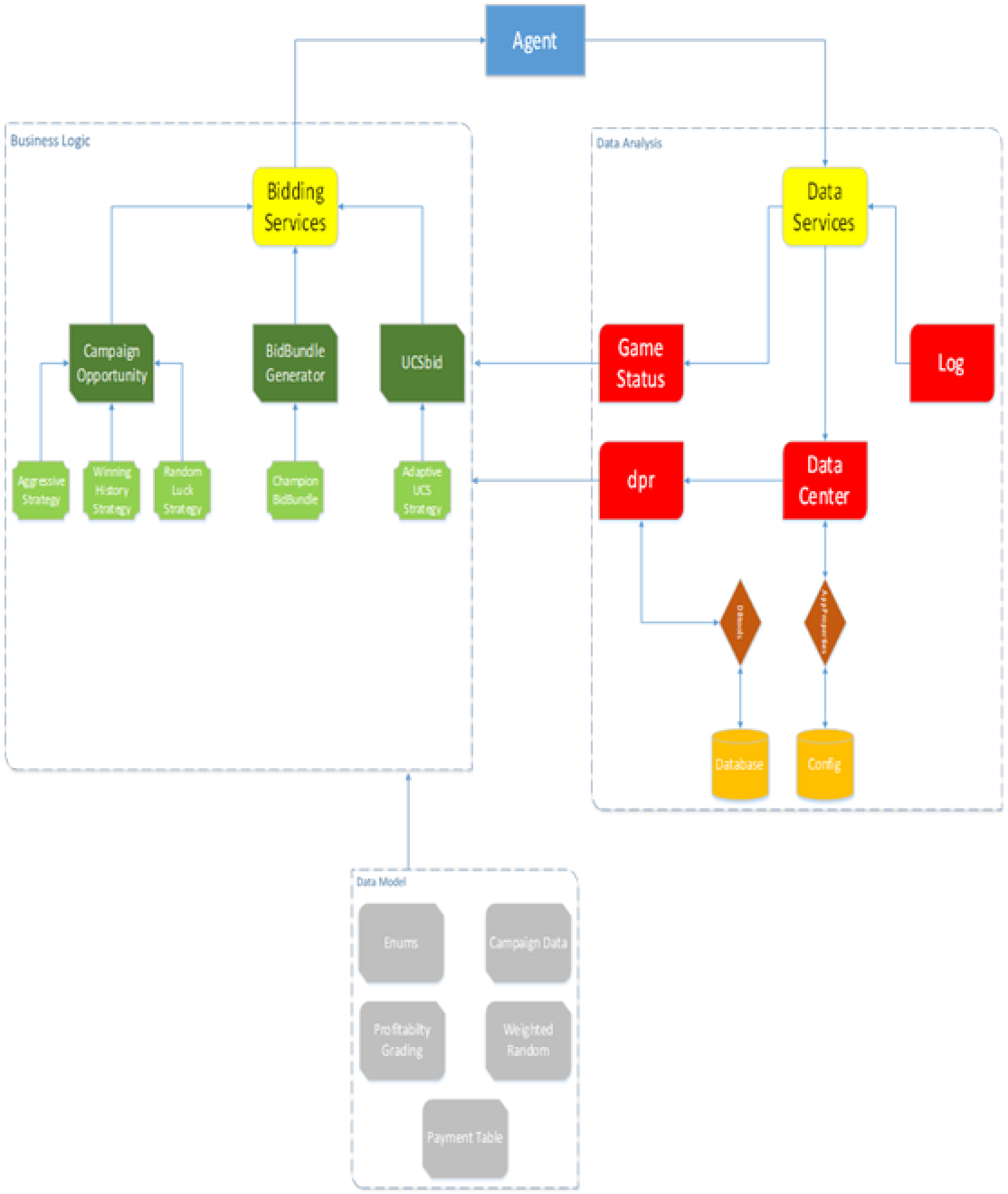
May 2014

1. Introduction

Our agent competes in a game that is constantly changing, where an agent has to apply different tactics and strategies to improve its own functionality but also to out-bid its opponents; where its evolution goes from “learning from its mistakes” to “anticipating the mistakes of others”; where the agent’s ability to win campaigns depends on its ability to fulfill them and vice versa.
All these properties mentioned above, alongside the fact that we were suffering from a maturing game server, are just examples of how extensive and thorough our work on Agent2 had to be.
Our work initiated in a research of the vickrey auctions that characterize the game, and of strategies, tactics and approaches used by bidders in the real world to out-bid their opponents and win.
We then decided of a game plan, which outlaid how we see the agent’s future performance, how will it act when its response is required, and most importantly how it will re-act to the game’s events when needed.
It is important to mention that this great deal of work and planning was done even before one line of code was actually written.
Our implementation of the agent’s focused on different strategies, each fitting a different status that the agent was in at the time, and each exploiting different game parameters, as given in the game daily reports. In addition, most strategies were dynamic, and kept making small adjustments to themselves at the end of each day, campaign, and game, based on its own and others’ performance.
We’ve decided that it would be best to start aggressively on all fronts, i.e. to bid highly on the first few campaigns, to acquire a satisfactory UCS level at the beginning, and to win most or all impression opportunities for the first few campaigns, in order to establish an advantage on other agents regarding the quality score.
We also included in our implementation event that may occur and were out of our hands, like the chance of losing the campaign even if we gave the best bid. We introduced different strategies to cope with these situations, and constantly adjusted them.
Our work is explained in detail in the following sections of this paper, as well as in the updated specification file. We start off with a detailed outlay of the strategies, what they consist and how they change during and after each game. We then elaborate on other features that we used to support the agent’s learning mechanism. Following that, we discuss the use of these strategies when bidding for campaign opportunities, and describe how UCS and impression bids were

calculated. We hope this paper, and the updated specification file, will provide the necessary information regarding our agent's implementation and performance.

2. Updated System Architecture



See full design specification document for details.

3. Methods used

3.1. Strategies Approach

Overcoming an opponent is a difficult task, especially when a large portion of the needed information is not available, or partially available at best. In order to cope with this problem, we came up with the strategies' approach. The purpose of the different strategies is to come up with a winning bid (for campaign opportunity, bid bundle or UCS), which is calculated using public game data and private learnt information. These calculations will be thoroughly reviewed later. Each decision of the game (campaign opportunity, bid bundle, UCS) is treated by a few *strategies* and a single controller who choose every day the appropriate strategy according to the current circumstances and previous successes and failures.

The strategy approach is composed of two major parts:

1. *Strategy Inheritance* – Our goal was to keep the creation and insertion of new strategies as simple as possible. Our agent operates within the boundaries of a game which is constantly changing, once the agent is in the lead and later it trails behind, and so we believed that it should possess a variety of strategies. This way the agent has a better chance of surprising the opponents and provide a winning bid. This has been implemented so that when a new strategy was introduced, all that had to be done was to inherit the Strategy or WeigthedStrategy classes.

2. *Weighted Random* – While having a variety of strategies sounds like a good plan, some of them must project better results than the others. We needed a way to distinguish the better strategies from the weaker ones. In order to do that we gave each strategy a "weight" value. The default weight was calculated as follows:

$$Default\ Weight = \frac{\text{SuccessRate}}{\text{Tries}} + \frac{\text{bid}}{\text{MaximumBid}}$$

Each Weighted Strategy had its weight value, which was adjusted during and between the games so the agent could learn which strategies are better and use the stronger ones more often.

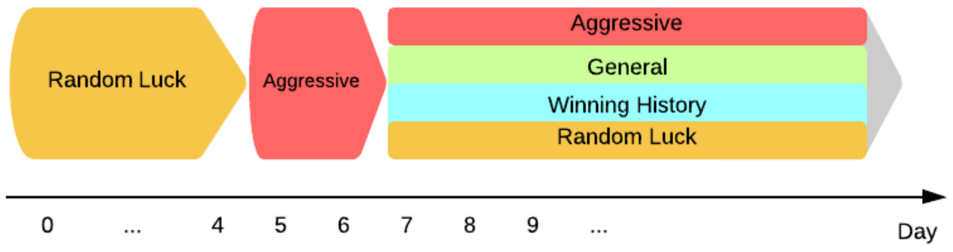
3.2. Inter-Games Learning

For the agent to evolve and become more efficient we need a way to save data between simulations. If we save the relevant data we can process it to improve the agent's performance. Furthermore, being able to easily configure and adjust the agent between simulations can be an important key for winning.
To easily store and process data we used SQLite database, to easily save and configure constants inter-games variables we used Java Properties file.

Inter-games data includes: strategies' ratio of success, winning bids per segment, multipliers for different campaign stages (concluded by the achieved campaigns reach), default UCS bid calculated, etc.

3.3. Campaign Opportunity Strategies

There are 4 major strategies in the campaign opportunity section: History Winnings, Random Luck, General and Aggressive. The goal, obviously, was to win as many campaigns as possible at the best price (and to get positive high profits).
First 6 days' strategies were preselected; in the rest of the game strategies are chosen by the weighted random mechanism described above.



3.3.1. History winnings: (*missing Auction Report)

We saved in the database bids of previously won campaign opportunities. Later, we used the database to get the best bid for the current campaign opportunity's segments. This was a very powerful strategy until Auction Report was revoked. This caused the relevance of the strategy to diminish as we were able to document only our own bidding successes. After that, the agent main strategy was to wage price wars against the other agents or hope for random luck campaign.

3.3.2. Random luck

Because the winning of the campaign opportunity isn't always related to best bid, the chance of winning a campaign at high cost could lead to high profits, therefore bidding the highest possible bid for a campaign and hope for random luck was chosen as a strategy.
First days of each game were found as the most difficult days to win impressions (due to the existence of 8 initial campaigns and the relatively aggressive behavior of agents on the first days). Therefore random luck was chosen to be the first four days strategy: running a few campaigns with low budget and fulfilling each campaign's reach on the competitive first days is almost an impossible mission in which failure will result in low quality score and a poor chance to win campaigns, but completing a fortunate high budget campaign is simple and gives us a significant advantage over other agents.

3.3.3. Aggressive

In order to be a competitive agent one most win campaigns and although there was some built-in “corruption” in the games, the best way to win a campaign is to offer the best bid possible. Therefore, aggressive strategy is offering the lowest possible bid to win a campaign. The strategy is able to make adjustments. For example, if our agent is winning multiple campaigns, raise the bid, and otherwise lower it.

3.3.4. General (*removed)

This strategy is composed of a general bid regarding the length of the campaign, and the reach, this strategy was abandoned because it gave poor results - the “real” agents unlike dummies submitted much lower bids, and if you are not bidding the best bid you may as well try random luck.
The general strategy formula:

$$\text{Max}\left\{0.65CampaignReach \cdot 0.1CampaignLength \cdot \frac{StrategyRatio}{QualityScore}, 0.1CampaignReach + 1\right\}$$

with strategy ratio determined by the strategy success.

3.4. UCS Bid Strategies

As "knowledge is power" in this game, we set our goal to get the highest UCS level, but also to avoid falling into a price war.
A configurable UCS Target level was defined – high enough to make sure that relying on impressions' probabilities is minimal, and not too high as fighting with another agent on the first place may be harmful to campaigns' profit. Trial and error led to a target of 80%.
During each game, prices of high service level (more than target level) are stored, and by the end of the game an average of the **median** of these prices and the default UCS bid from previous games is saved as the new default UCS bid.

$$DefaultUcsBid_{g+1} = \frac{DefaultUcsBid_g + Md(\{UcsPrice_{g,d} | UcsLevel_{g,d} \geq 0.8\})}{2}$$

First 3 days strategy is to bid higher than the default UCS bid (configurable multiplier), as getting a high Quality Score in the first days helps us to win campaigns and start in a position of strength.

$$for\ d \in \{0,1,2\} \quad UCS\ Bid_{g,d} = 1.2 \cdot DefaultUcsBid_g$$

Day 3 and beyond strategy is the Adaptive UCS Strategy, unless 5 days in a row with low UCS level (< 0.6) and at least one active campaign has passed – in that case an aggressive one-day-strategy of giving the maximum bid (as configured - 0.8) is applied.

3.4.1. Adaptive UCS Strategy

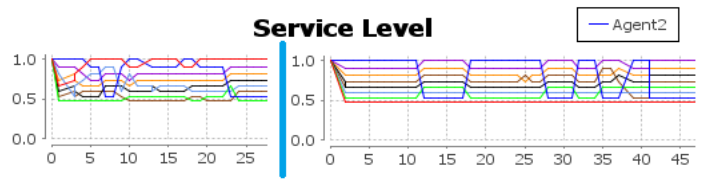
The Adaptive UCS Strategy, which runs during most of the days, reacts to the changes in UCS level and prices.
Initial bid is the default UCS bid (described above). On the next day, the response depends on the outcome: if the current UCS level is higher than the UCS target (described above), we

carefully try to reduce the cost – by getting closer to the second bid (average of bid & price + epsilon), or at least by subtracting a configurable minimum step.
If the current UCS level is lower than the UCS target, we increase the bid by a configurable factor.
The bid considers a number of other factors:

- Number of active campaigns: the more campaigns currently active, the greater the benefit from a higher UCS level. If there are zero active campaigns, any charge on UCS is unnecessary.
  Therefore, the bid should be an increasing function of the number of active campaign. Multiplying the bid by a cube root of the number of active campaign should do it. (Square root was found growing too fast.)
- Price elasticity: defined

$$E_{UCS} = \frac{\text{UCS Price}}{\text{UCS Level}} \cdot \frac{\Delta\text{UCS Level}}{\Delta\text{UCS Price}}$$

  this phrase (calculated by the last 2 days) gives the percentage change in UCS level in response to a one percent change in price. If $|E| < 0.5$ (inelastic), a 1% change of UCS price generates less than 0.5% change in UCS level and increasing the price is not productive. If $|E| > 2$ (elastic), a 1% change of UCS price generates more than 2% change in UCS level and decreasing the price is not productive.
- Minimum & Maximum bids – to avoid unexpected changes.



Typical service level charts for the adaptive strategy (from first demos)
on the left: 0.85 target level
on the right: 0.97 target level
notice the low level (and price) on 0 active campaigns days

3.5. Bid Bundle Strategies

In order to win the game, an agent must be able to increase its profits not only by winning campaign opportunities, but also by fulfilling these contracts to the fullest. By doing so, the agent will earn the highest portion of the agreed upon campaign budget, as well as obtain the highest quality score, which is of top importance to future campaign bidding.

3.5.1 Champion Strategy

The strategy's sole responsibility is to generate proper and hopefully winning bids for available impression opportunities.
The strategy relies on three probabilistic variables when computing these bids:

1. Each campaign's *Current Impression Acquisition Level* (CIAL):

$$CIAL = \frac{\text{quantity of impressions acquired}}{\text{total agreed upon impression reach}}$$

2. Each campaign’s *Previous Day Success Level* (PDSL):

$$PDSL = \frac{\text{quantity of impressions acquired on the past day}}{\text{total agreed upon impression limit on the past day}}$$

3. Each campaign’s *Overall Success Level* (OSL):

$$OSL_t = OSL_{t-1} * \frac{days\ passed\ since\ campaign\ start}{days\ passed\ since\ campaign\ start + 1} + PDSL_t * \frac{1}{days\ passed\ since\ campaign\ start + 1}$$

After calculating these three variables, the strategy turns to a *Payment Table*, which consists of pre-defined intervals, all between 0 and 1. The table’s keys are interval arrays of length 3, i.e. {interval for CIAL, interval for PDSL, interval for OSL), while the values are coefficients, to be used in future bid calculations. When addressing the table, the strategy verifies when all three variables fall into the specific interval permutation key in the table, and then and only then returns the correct coefficient.
It should be mentioned that these coefficients are the results of on-going trial and error, and were updated constantly in the configuration file, in order to obtain maximal impression reach with minimal expenses.

After obtaining the correct coefficient, this coefficient is multiplied by three multipliers: *Day Multiplier* (DM), *Config Multiplier* (CM) and *Game Multiplier* (GM). These values are updated dynamically and manually in the configuration file.
The DM is a value which depends the number of days left for the current campaign to run out (as a percentage of the whole campaign). If the campaign has only just begun, the value would be relatively low, and will be increased as the campaign will be nearing its end. The basic notion behind this is that the agent should send higher bids when there is not much time left, in order to try and obtain as much impressions as possible. On each simulation end, the three possible values (low, mid and high) are enlarged or reduced by the total impression reach of campaigns in the game.
The CM is fixed value which is updated in the configuration file between runs in order to improve results.
The GM is a value which depends on the number of on-going campaigns at the moment, not only these which are handled by our agent but all of the campaigns handled by all participating agents. The GM is calculated as the number of currently on-going campaigns (via campaign opportunities) divided by the number of an “average” day’s on-going campaigns (5).

We consider this value because it represents a “difficulty” level for winning impressions. If many campaigns are active at the moment, more agents will compete for the same impression opportunities, and so higher bids are required in order to win them. On the contrary, if only several campaigns are currently active, a lower bid might still be sufficient in order to win impressions.

Finally, the calculated bid serves as a coefficient, which is multiplied by the budget of the campaign, and is then divided by the impression reach. This is done in order to “normalize” the bid. By dividing the budget by the reach, we obtain the price per impression which will generate a balanced, not winning and not losing revenue. By multiplying this value by a bid we previously calculated, we are considering other elements of the game, ours and our opponents, and reflect them in a final bid, which we expect to generate maximal acquisition of impressions, resulting in higher revenues.

In conclusion, the bid will look as follows:

$$bid = \mathrm{PyamentTable(CIAL, PDSL, OSL)} * \mathrm{DM} * \mathrm{CM} * \mathrm{GM} * \frac{Campaign\ Budget}{Campaign\ Reach}$$

3.5.2 First Campaign Strategy

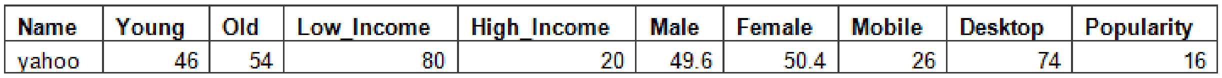
Because the first couple of days of the simulation have more active campaigns running simultaneously we used higher bids to win more impressions and maintain high quality score. On each simulation end, a specific multiplier is calculated for the first days, based on the achieved impressions reach of first 6 days’ campaigns.

3.5.3 Unknown User Classification

Up till now we considered bids which were calculated and sent according to each campaign’s unique attributes, i.e. device and ad-type coefficients, market segment etc. However, the ability to map each publisher’s website and to match it with the campaign’s current goals depends greatly on our knowledge, which in turn depends on how well we bid for our knowledge level, i.e. our UCS level. The calculation of the latter bid is thoroughly discussed in a previous section of this document, and will not be repeated here. However, one should be prepared for the scenario where the obtained UCS level is poor, thus making the bid bundle calculation work much harder. In such a case, our agent should consider submitting bids to publishers which it contains little or no information about their available impressions.
In order to do so without resulting in over-bidding or winning non-target impressions (which have no value to the advertiser and so no revenue comes out of them), our agent sends these “unknown” bids only to publishers which were given the highest grade using its *Grading Mechanism*. These publishers were stored in the *Profitability Grading*. The number of publishers

which received these bids depended on the current UCS level. If the agent's UCS level was high, i.e. it had extensive knowledge regarding users' distribution, an "unknown" bid was sent only to a few top publishers, in order to enhance profitability to the fullest. However, if the agent's UCS level was poor, the agent had basically nothing to lose and so it "shot in all directions" hoping to enhance profits by obtaining relevant impressions.

Following is a brief review of the *Profitability Grading* and the *Grading Mechanism*:
We saved the Publisher Table (based on Table 3 from the game's specification file) in the database that we established for the game. This table consisted of several publishers and their relevant statistics, including the distribution of the users who visit this publisher site to each market segment, each device used to enter the site and the popularity of each site amongst the user society of the game. For example, a record in the table:



Name	Young	Old	Low_Income	High_Income	Male	Female	Mobile	Desktop	Popularity
yahoo	46	54	80	20	49.6	50.4	26	74	16

The ProfitabilityGrading class consisted of a map called *Grading Map.* This map had the agent's current active campaigns as keys, while the value was another map, called *PubAdDevice Map*. The second map had a 3-sized tuple as key, where each tuple was a permutation of a publisher, an ad-type and a device type (resulting in a total of 6*2*2 = 24 permutations), and the value was the grade of the tuple, for the specific campaign. For example, a record in *Grading Map*:

GradingMap(Campaign c, ([ebay, video, desktop], 32.3))

The grade was calculated in the *Grading Mechanism* as follows:
For each specific campaign, we traversed all the publishers in the game, and for each one we obtained all the data from the database table mentioned earlier, including statistics for the relevant segments of the current campaign. We multiplied all the parameters, and then multiplied the result by the statistics for each device usage and by each publisher's up-to-date ad-type orientation, as given in the most recent report. Finally, we introduced the popularity factor to the equation. In conclusion, the calculation was as follows:

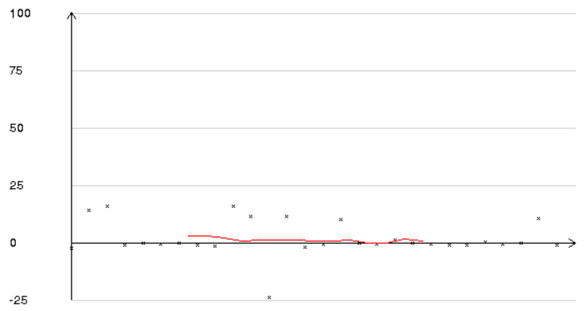
$$Grade = \mathrm{CampaignSegmentsStats} * \mathrm{AdTypeOrientation} * \mathrm{DeviceStats}$$

$$Grade = X * Grade + (1 - X) * popularity$$

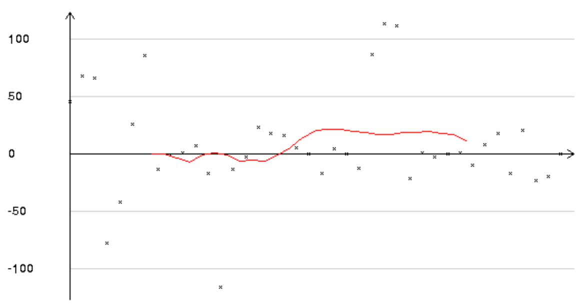
Where X was chosen to equal 0.7 after running the game many times, proving to be the most productive coefficient.

4. Performance & Analysis

Running our agent against dummy agents was quite different than running it against “real” agents in terms of performance.
During most competitive days, especially when the product (advertising) is homogeneous (similar quality scores), the game tended to illustrate a Bertrand competition with a Nash equilibrium where the winning price is equal to the marginal cost (and sometimes even lower, as some of the agents behaved irrationally).
It is important to mention that our agent’s performance was greatly affected by various exogenous factors. While many of these factors, such as server failures and changes in impression quantity and bidding affected our opponents as well, other factors had great negative influence solely on our agent’s performance, and required maneuvering and adjusting our code to fit these factors.
For example, the AdAuctionReport had an important role in calculating and running our Winning History Strategy, and its removal turned this strategy to practically useless.
Demo competitions - During the first competitions we encountered some very poor results (partly due to the recent changes in the game, and partly due to the performance of the rivals which we didn’t anticipate. Nonetheless, as these competitions progressed, we learned from the results and from demo to demo we improved our agent, until winning the last demo before the international contest.



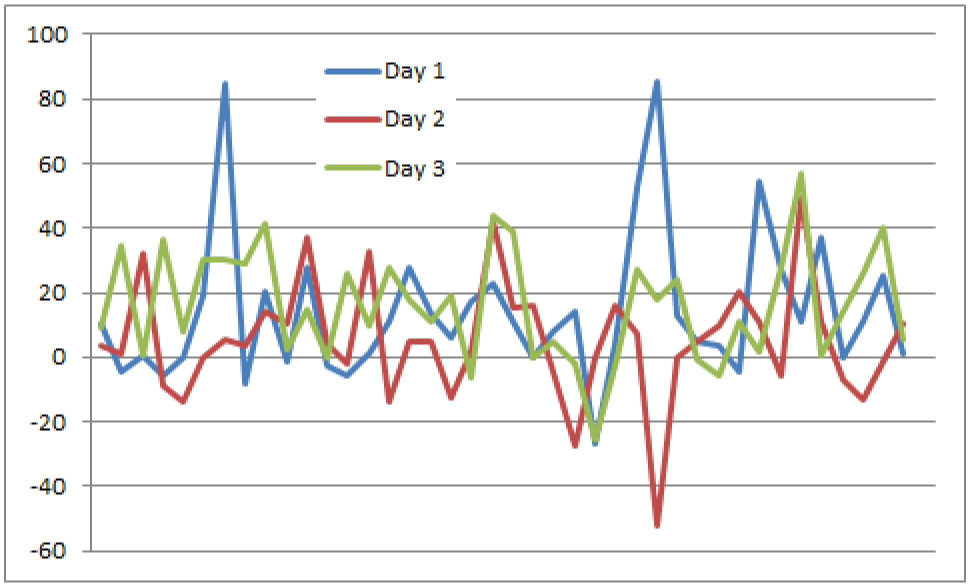
29/04 Demo competition



04/05 - Major improvement

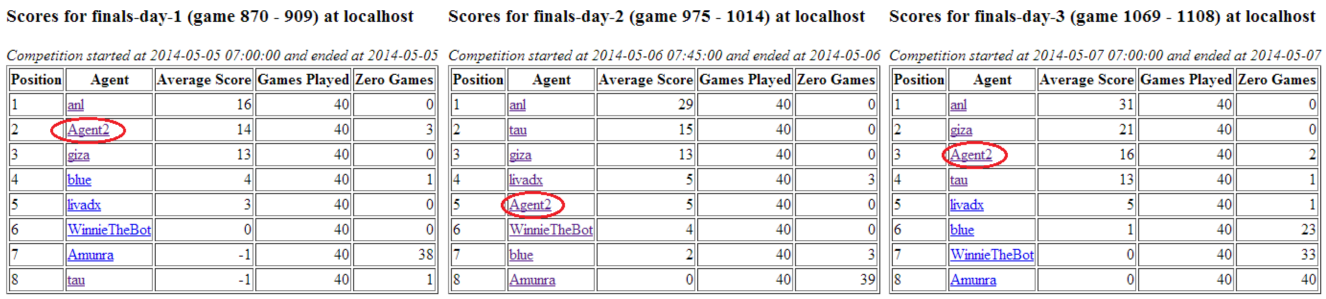
Participating in the international contest was an interesting and challenging experience. We had a chance to watch agents with different behaviors (especially the winning Chinese team, who move from rational to crazy random behavior and back), and understand how different the game can be depending on its participants. We found that Giza and us are playing the game with close strategies that sometimes weakened both our agents and let other rivals take advantage of that, and that in some days it’s better to be more “gentle” and not to get into a “bidding war” that weakens our reputation (quality score).

The first day of the international contest was more of a learning day, in which we learned how to deal with this new kind of competition. We did reach some nice results that day, and ended in the dignified second place.
Second day was much difficult due to some serious technical problems, and the third day was a tough competition with the BCM agent, ended in the third place that day.



Agent results through the games in TAC finals

TAC finals competitions scores



Scores for finals-day-1 (game 870 - 909) at localhost

Competition started at 2014-05-05 07:00:00 and ended at 2014-05-05

Position	Agent	Average Score	Games Played	Zero Games
1	anl	16	40	0
2	Agent2	14	40	3
3	giza	13	40	0
4	blue	4	40	1
5	livadx	3	40	0
6	WinnieTheBot	0	40	0
7	Amunra	-1	40	38
8	tau	-1	40	1

Scores for finals-day-2 (game 975 - 1014) at localhost

Competition started at 2014-05-06 07:45:00 and ended at 2014-05-06

Position	Agent	Average Score	Games Played	Zero Games
1	anl	29	40	0
2	tau	15	40	0
3	giza	13	40	0
4	livadx	5	40	3
5	Agent2	5	40	0
6	WinnieTheBot	4	40	0
7	blue	2	40	3
8	Amunra	0	40	39

Scores for finals-day-3 (game 1069 - 1108) at localhost

Competition started at 2014-05-07 07:00:00 and ended at 2014-05-07

Position	Agent	Average Score	Games Played	Zero Games
1	anl	31	40	0
2	giza	21	40	0
3	Agent2	16	40	2
4	tau	13	40	1
5	livadx	5	40	1
6	blue	1	40	23
7	WinnieTheBot	0	40	33
8	Amunra	0	40	40

We learned that a big part of an agent’s success is the survival of the early stages of the game with decent quality ratings so that in later parts of the game we can use the built-in “corruption” (random campaign allocation) to make good profit.
Another thing this “corruption” can help with is recovering from a bad start and improving ratings. The most important part is having sufficient quality ratings in order to place a valid bid, because if you can’t do that then you are out of the game.
In addition, inter-simulation improvement can be a major key in winning. Even a small improvement to the bid bundle spending’s can lead into a major performance improvement, as we fight over impressions. If the improvements are done correctly this can lead to winning more

impressions and pushing competing agents out of the equation. On the other hand, it can cause other agents to bankrupt while trying to buy impressions at a very high price.
We also learned that we should consider the campaign opportunity target segment before bidding on it, because competing against an aggressive opponent on the same market segments can lead to considerable loss of money and quality ratings.

5. Conclusions

We discovered that in order to obtain optimal results, our agent must be aggressive when it comes to acquiring and maintaining a high quality rating. This is mainly important in the final phases of the game, where there are only several agents at most with a sufficient quality rating, and so our agent will have an advantage over others and will be able to win campaigns more easily, and with better pricing.
In addition, the built-in “corruption” must be considered, because if there wasn’t one, the game would have turned to a competitive equilibrium, i.e. a zero-sum game. While implementing the agent, we tried to use this “corruption” to our advantage.
As we’ve seen during the competitions, every agent can come up with different strategies and tactics in order to overcome its opponents and so our agent had to be flexible and be able to learn from other agents’ behavior in order to improve its own strategies.
We’ve tried to make our agent as successful as possible by manually updating values and stats according to the results we’ve seen, as well as allowing our agent to update various values automatically.
As mentioned before, the work on the agent had been very thorough, and after countless sessions against dummy and real opponents we managed to reach a level which satisfied us. Although at times we felt that the work on the agent was reaching a dead end, we were able to overcome our problems and adjust our agent according to the changes made in the game.
In conclusion, the work done in this workshop, which included planning, designing and implementing an agent practically from scratch, was a fulfilling and important one.
We hope that this paper and the entire work done by our team throughout the semester will demonstrate how dedicated we were to the agent’s success.