Ad Exchange Workshop

Project Report

'BCM' Team

Gal Rotem	201313533
Lior Abadi	301139234
Tal Brender	200088300

Project Report- ‘BCM’ Team

As the background and broad outlines of the ADX game have been given in the introduction of the spec draft, we wish to focus in this document on expounding the development process of our agent with respect to the general strategies and motivations that underpin it. This depiction will be supported by an exemplary analysis of competitions we participated in backed by results.

Absolute vs. Relative value

At the beginning, we faced the task of delivering a decent agent that can win a competition against dummy agents. The code’s initial algorithms revolved around naive calculations of mean values. Without any true knowledge of the game’s behaviour, we added minor tweaks to these calculations to test the response of the game. For instance, we added conditions that limited bids for campaigns within a certain range of days. Of course, these tweaks were of a trivial nature, but they sufficed for that point of time.

This experience lent us a basic but useful intuition about the game: tweaking values within a reasonable range of values upon learnt experience improves results. The word “tweaking” emphasizes a crucial point here: sometimes the outcome of using a certain value (for instance an initial factor we use, e.g. our so called “smooth factor”) is rather obscure, and can only be understood with time, meaning with playing against other agents. Of course, deciding that we **need a certain factor** (like the aforementioned smooth factor) comes only after understanding absolute notions about the game. So we come upon an important distinction between deciding that we need a certain value and choosing an initial value and tweaking with it (within reasonable bounds). This distinction can be rephrased as holding between two important elements of this game (and surely other games with similar traits) : **absolute value** and **relative value**.

Absolute value: this sort of value reflects aspects of the game that we can rely on throughout the game. An absolute value can be assigned to a certain behaviour we expect to improve gradually or at least act steadily. It can also be assigned to static statistics we were given before running the game (plus dynamic statistics such as the popularity of websites during the game) . Upon identification of a so called absolute value, we devised a certain formula that reflects this value. Of course, we needed to give weight to factors that are susceptible to changes along the game, thus balancing the “absoluteness” of these values with relative and somewhat random changes.

Relative value: this sort of value reflects aspects of the game that we can evaluate only upon gradually learning behaviours throughout a given game or between games (meaning, with time). A relative value is assigned to a certain aspect of the game after realizing that it behaves in a way that is hard to predict. For instance, a relative value is assigned to initial values of factors, such as the so called constant UCS_INTIAL_SMOOTH_CHANGE_FACTOR. The initial value we chose for this constant couldn’t be calculated by a formula, the output of which is known to reflect an absolute value. On the contrary: like other factors we used, and like learning in general, initial values are always infected with an unknown nature that could be only learned with

experience. Thus, after playing games against different versions (executables) of our agents, we decided to assign: UCS_INTIAL_SMOOTH_CHANGE_FACTOR = 0.3. Of course, incorporating this factor into the ucs bid algorithm came upon realizing that improvement can be done only against our past behaviour, coupled with a belief that reusing it along time will grant us with improving results. Thus, this usage harnesses both absolute values (gradual improvement, naive linearity between subsequent bids) and relative ones (rough linearity should be smoothed with time, picking an initial value via experience).

One example of identifying an absolute value and devising a relevant formula is the definition of "activity volume" with respect to the UCS bid algorithm. As explained in the spec draft, this concept reflects the amount of targeted impressions we wish to win in the following day. We know that this factor is crucial while we consider a UCS bid: the higher the UCS rank, the more targeted impressions we expect to win. The activity volume can be calculated upon information we reserve throughout the game with respect to an agent: the duration of a certain campaign, the reach of that campaign, an assessment of how much impressions the agent won until a certain point of time (relative to the duration of the campaign) . Upon defining this concept, we pinpointed two important factors the define this value:
RDTG, which is the amount of days left relative to the campaign's length; RITG , which is the impressions left to suffice the campaign relative to its initial reach.
We searched for a function that describes the following behaviour: the activity volume for a campaign surges as the campaign approaches it's end, and plummets as the number of impressions-to-go decreases. Then, we drew this behaviour on a piece of paper and searched for a function that has a similar graph. The output was $\sqrt{(1 - RDTG^2) * RITG}$, the graph of which is given in the spec draft.
In this example we can see a definition of an absolute value, one that we can rely upon its usage in the game in favour of better results.

As we improved our agent, we identified the importance of keeping the quality rating within a reasonable range of values. During one of the first unofficial competitions against other agents from the workshop, we noticed how calamitous could a QR drop be for a certain game. Failing to satisfy the initial campaign properly, we experienced a major crash of the QR value, which simply prevented us from winning campaigns during the games. We came to understand that there will always be some unpredicted behaviour lurking around the corner, so the best we can do is to react quickly to such oscillations, in order to minimize harms. After noticing the exponential behaviour of the ERR function, we devised a formula (which can be seen in the spec) that incorporates the QR into the calculation of the campaign bid epsilon.

This sort of reaction to QR oscillations reflects our continuous attempt to tame "relative valued" behaviour with an absolute value insight that is manifested through a formula. Again, we realized that randomness cannot be predicted (by definition) and all we could do is react properly.

Another identification of absolute value resulted in usage of dynamic and static statistics. Of course, prediction of future behaviour is corroborated by usage of past experience. Some of the statistics we used were of the static sort: for example, those concerning the probability of a certain user to enter a certain website. Together with dynamic statistics regarding the popularity of websites along the game (as given by usage of the function handleAdxPublisherReport), we devised a formula (using the formula given in the spec of the ADX game) to calculate the chance of a user from a triple market segment to attend a website. The outcome of this formula is used to calibrate the actual bid give for non targeted impressions. This usage can be seen in the bid bundle algorithm.
Another usage of dynamic statistics is manifested in the saturation level, which is explained in detail in the spec. This rate signifies the saturation of a certain market segment, a value that can be evaluated upon saving the available data on rivals (we know the campaigns and their market segments that were won by a certain rival).
This sort of information can be deemed as “absolute” mainly because it reflects aspects that are less given to oscillations: bids for campaigns with market segments that are saturated should be higher priced, because sufficing them would be harder.

Methods of work

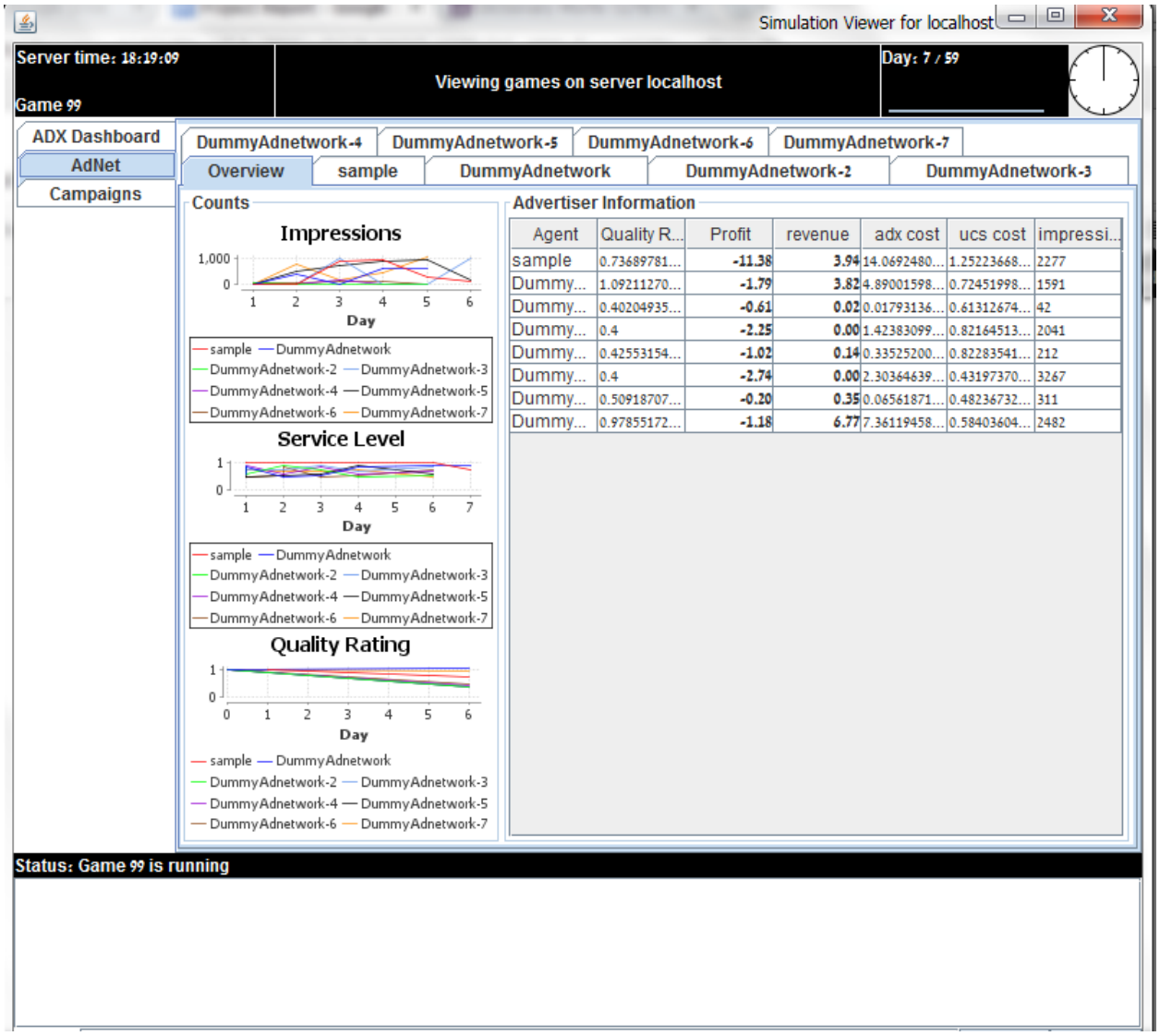
The source code was placed in an SVN remote repository. This enabled us to work separately on the code after dividing the work among the team members: One focused on statistics processing via usage of different utility functions , the other focused on the UCS bid algorithm and shared with another work on the bid bundle algorithm and campaign bid algorithm.
At the beginning, we improved our agent via playing against dummy agents and tweaking the initial naive calculations. Then, we defined important absolute values and devised formulas that combine them with relative values in a trial and error process.
The leap in our agent’s ability came with competing with agents from the workshop. This way we identified inherent weaknesses of our code and understood what strategies of other agents are beneficial for us, and what strategies are harmless. Upon recognizing a weakness (an example of tackling one is given above with the QR issue), we inserted changes in the relevant places and tried them against the other agents and against older versions of our agent.
Among issues we would have liked to improve is the prospect of using data between different games, perhaps by saving it to external files that could be used among games. Playing in the international event, we noticed that some agents were improving steadily from a given point of time, perhaps suggesting that they had easy mechanisms to change quickly important factors from game to game. Of course, we inserted tweaks between games, but perhaps a more automatic functionality for controlling bounds and factors could have served us better.
Nonetheless, we feel that our agent became better with time and that our aggregated experience could serve future agents and perhaps a better version of ours.

Game Analysis

We held a demo competition against dummy agents, we shall analyze and describe how our agent is adapting throughout the game, and his strategy.

Our agent sees the first 5 days of the game as “dangerous” days, since we expect the market to be very saturated, we are willing to lose on the first campaign, in order to have a higher quality rating than our rivals, in such a scenario, we expect to recover our losses quickly, because we should be able to win campaigns more easily at a profitable price.

After 5 first days of the game :

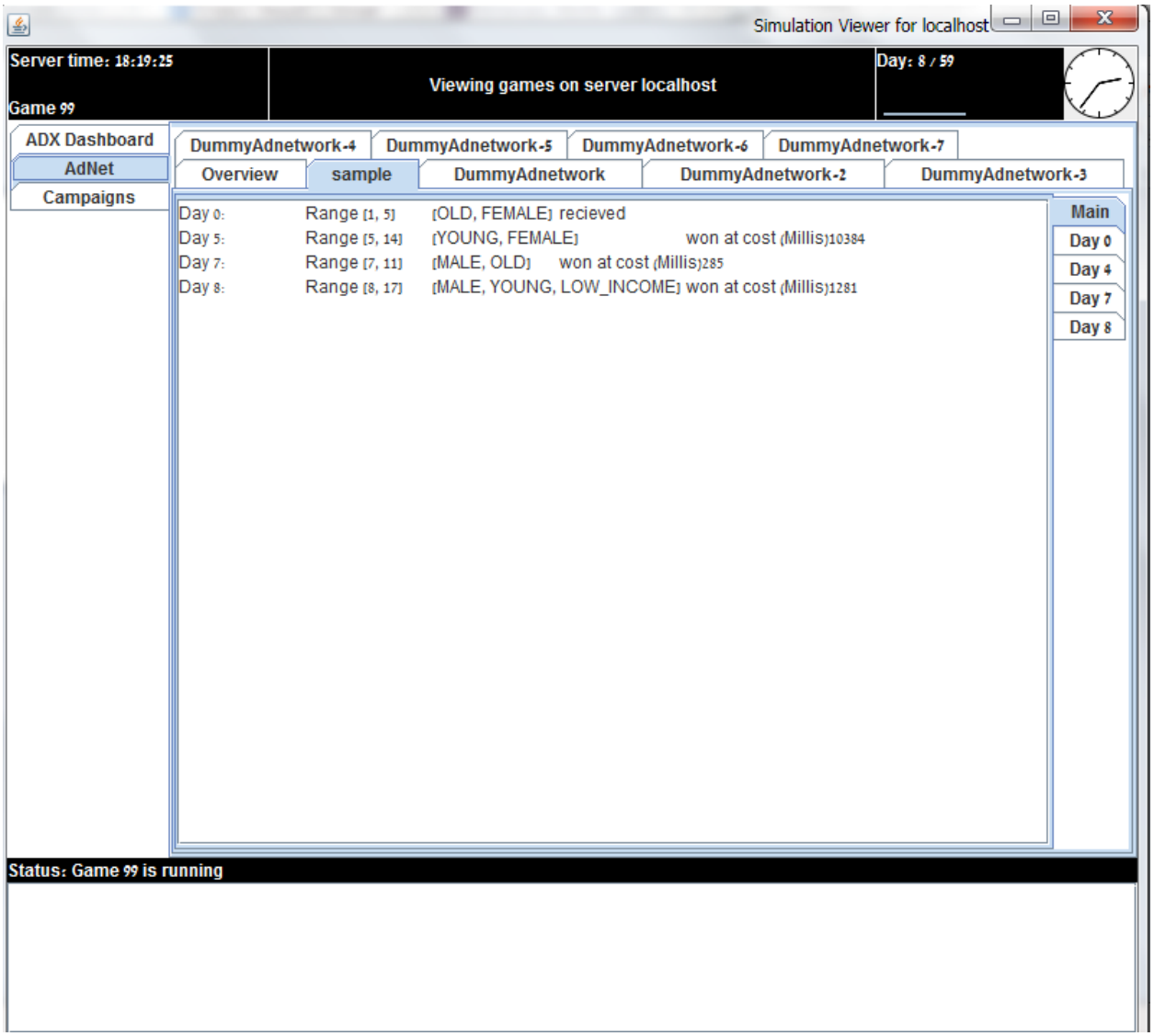


Agent	Quality R...	Profit	revenue	adx cost	ucs cost	impressi...
sample	0.73689781...	-11.38	3.94	14.0692480...	1.25223668...	2277
Dummy...	1.09211270...	-1.79	3.82	4.89001598...	0.72451998...	1591
Dummy...	0.40204935...	-0.61	0.02	0.01793136...	0.61312674...	42
Dummy...	0.4	-2.25	0.00	1.42383099...	0.82164513...	2041
Dummy...	0.42553154...	-1.02	0.14	0.33525200...	0.82283541...	212
Dummy...	0.4	-2.74	0.00	2.30364639...	0.43197370...	3267
Dummy...	0.50918707...	-0.20	0.35	0.06561871...	0.48236732...	311
Dummy...	0.97855172...	-1.18	6.77	7.36119458...	0.58403604...	2482

We can note that our profit is poor, however, our quality rating is the 3rd highest, hence, we should be able to win campaigns and recover both our quality rating and our profit.

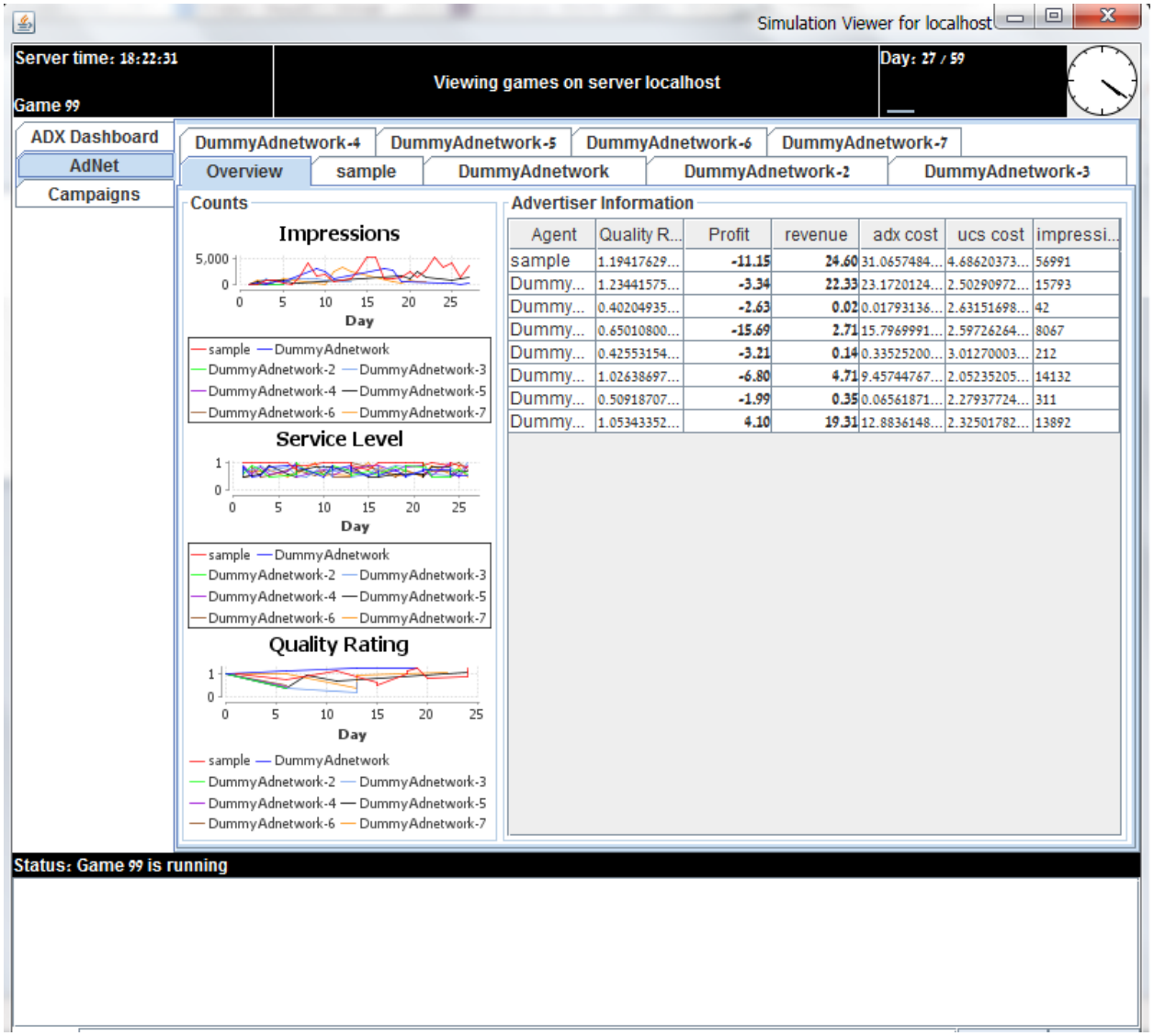
Our adx cost was relatively high, because the initial campaign we had intersected with our rival campaigns, bidding higher was necessary in order to win impressions and have a decent quality rating.

In our strategy, in order to improve our quality rating after the first 5 “dangerous” days, we will bid lower for campaigns in the following days, this action is taken in order to quickly boost our quality rating. In these following days , we expect the market saturation level to be lower, making our campaign targets easier to achieve, this strategy is well noted in the continuation of the above game:



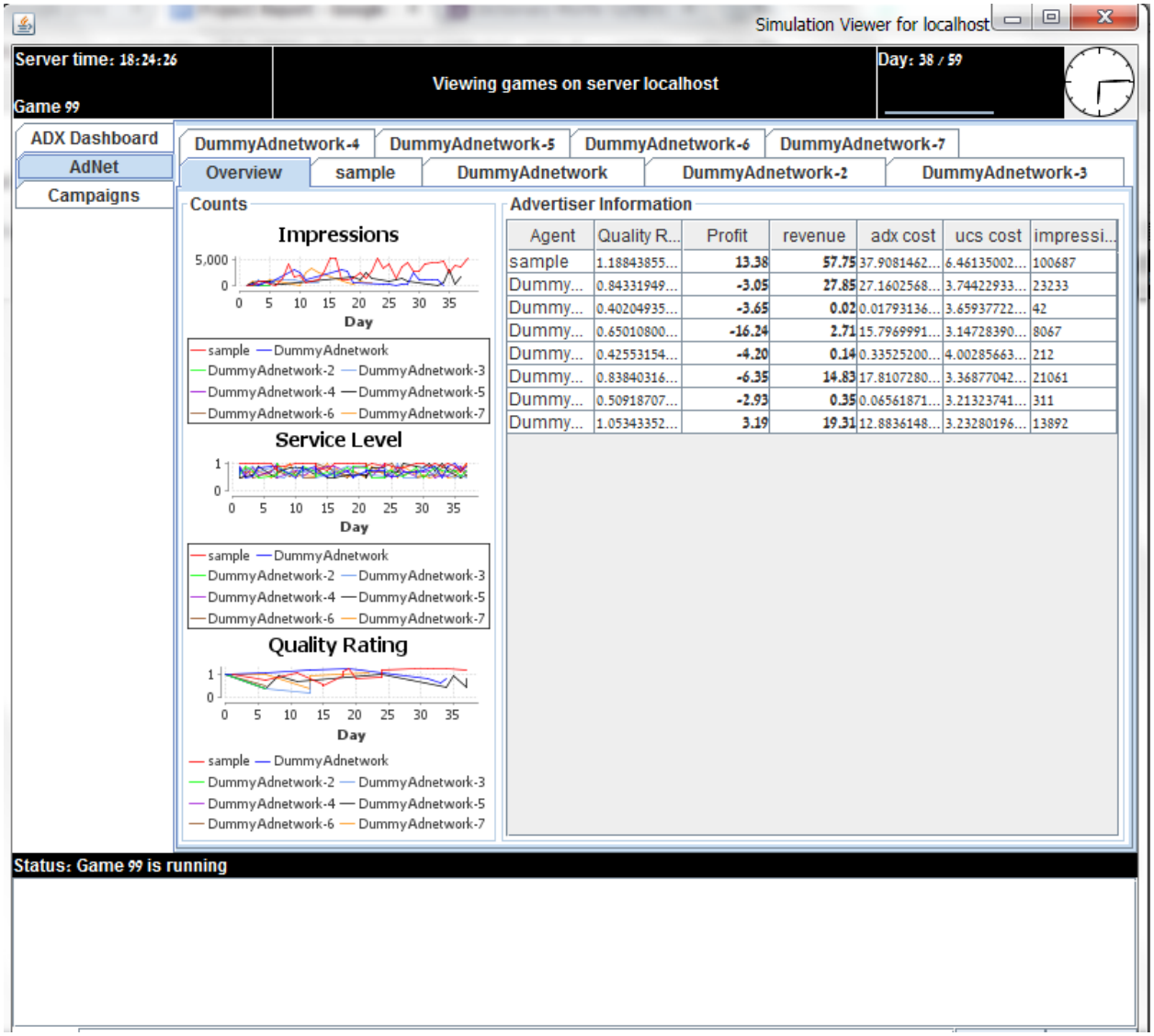
Notice that immediately when the initial campaigns end, we bid lower , and win the offered campaigns in these days.

We then continue with the strategy explained in the game spec, we can clearly see that our quality rating is improving and after 27 days we have the second highest quality rating:



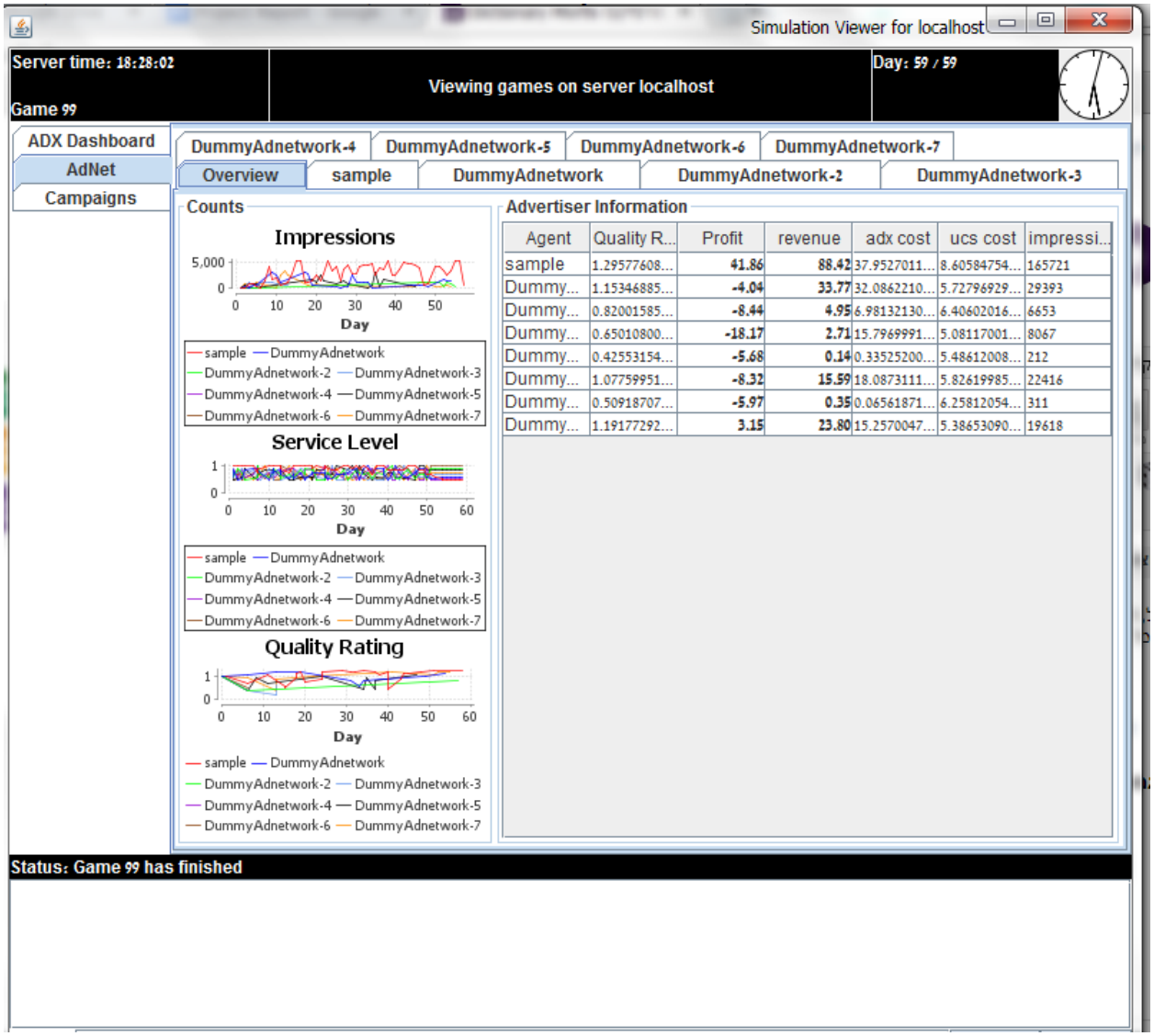
Building a good quality rating is important, since having higher quality rating than our rivals allow us to bid a more profitable bid for campaigns, meaning that each campaign we win will be profitable to us. Having said that, we can still see that our profit is negative, while approaching nearly day 30, we are aware that our strategy may be dangerous at some individual games, but we believe that in total, our agent will finish his 60 days with a positive and a relatively high profit.

As the game progresses, we recover from the losses we had in the beginning of the game, by winning campaigns at a profitable price:



Agent	Quality R...	Profit	revenue	adx cost	ucs cost	impressi...
sample	1.18843855...	13.38	57.75	37.9081462...	6.46135002...	100687
Dummy...	0.84331949...	-3.05	27.85	27.1602568...	3.74422933...	23233
Dummy...	0.40204935...	-3.65	0.02	0.01793136...	3.65937722...	42
Dummy...	0.65010800...	-16.24	2.71	15.7969991...	3.14728390...	8067
Dummy...	0.42553154...	-4.20	0.14	0.33525200...	4.00285663...	212
Dummy...	0.83840316...	-6.35	14.83	17.8107280...	3.36877042...	21061
Dummy...	0.50918707...	-2.93	0.35	0.06561871...	3.21323741...	311
Dummy...	1.05343352...	3.19	19.31	12.8836148...	3.23280196...	13892

Here, we have the highest quality rating and the highest profit, we continue to bid higher for more saturated market segments , along with having a high quality rating in comparison to our rivals, we can win campaigns at a profitable price, and win all targeted impressions they require, finally , the game ends with the final score:



Agent	Quality R...	Profit	revenue	adx cost	ucs cost	impressi...
sample	1.29577608...	41.86	88.42	37.9527011...	8.60584754...	165721
Dummy...	1.15346885...	-4.04	33.77	32.0862210...	5.72796929...	29393
Dummy...	0.82001585...	-8.44	4.95	6.98132130...	6.40602016...	6653
Dummy...	0.65010800...	-18.17	2.71	15.7969991...	5.08117001...	8067
Dummy...	0.42553154...	-5.68	0.14	0.33525200...	5.48612008...	212
Dummy...	1.07759951...	-8.32	15.59	18.0873111...	5.82619985...	22416
Dummy...	0.50918707...	-5.97	0.35	0.06561871...	6.25812054...	311
Dummy...	1.19177292...	3.15	23.80	15.2570047...	5.38653090...	19618

We do see that our adx cost was pretty high (almost half of our revenues), but we maintain this strategy because we believe that higher quality rating will eventually result in higher profits, as the game progresses.

Our strategy is working well not only against dummy agents, but also against real agents, implemented by other groups, as we can see in these games, which were held:

Position	Agent	Average Score	Games Played	Zero Games
1	Agent2	7	40	3
2	BCM	3	40	0
3	ibm	3	40	1
4	agent00	0	40	40
5	sample	-4	40	0
6	OOS	-5	40	9
7	giza	-12	40	1

In the next games we played under the name ‘tau’ (international competition):

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

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 note that throughout the competition, our agent was stable, keeping his place in the higher part of the ‘table’.