TAC AdX'14: Autonomous Agents for Realtime Ad Exchange*

Bingyang Tao, Fan Wu†, and Guihai Chen
Shanghai Key Laboratory of Scalable Computing and Systems
Department of Computer Science and Engineering
Shanghai Jiao Tong University, China
sunicy@sjtu.edu.cn, {fwu, gchen}@cs.sjtu.edu.cn

ABSTRACT

Ad exchange, as a new advertising trading form, has been widely applied nowadays. It is an automatic marketplace requiring autonomous ad networks to bid for ad slots on web pages in realtime. The Trading Agent Competition Ad Exchange (TAC AdX) is a new challenging competition for agents who focus on the autonomous ad trading marketplace and the competing strategies on behalf of ad networks. Specifically, agents are required to compete for adverting contracts and to fulfill them in the ad exchange. In this paper, we present our agent, namely *ANL*, who won the first TAC AdX, with a total profit 1.592 times that of the runner up. We further propose a more adaptive and active agent, called *AdvANL*, which performs even better than *ANL* in our simulated competition environment.

Categories and Subject Descriptors

I.2 [**Computing Methods**]: Artificial Intelligence

General Terms

Algorithms; Experimentation; Economics

Keywords

Trading agent competition; ad exchange

1. INTRODUCTION

Online display advertising market has grown significantly in the last decade. It generates about 42.8 billion dollars in the US in 2013 [7]. Advertisers used to sign advertising contracts negotiated with either publishers (website owner) directly or through ad networks.

As the market booms, an efficient way for trading advertising opportunities, called ad exchange, has been introduced recently [8], such as Google DoubleClick [1], Yahoo! Right Media [3] and Microsoft Ad Exchange [2]. The ad exchange provides publishers and advertisers with an open and convenient marketplace, which potentially increases mutual benefits. First, publishers can expect a better price than that from a single ad network/advertiser. Second, advertisers may sign advertising contracts with less budgets. Whenever a user visits a publisher's web page, the ad exchange conducts an auction for the ad slot among relevant ads from ad networks[1]. The winning ad is then displayed and the corresponding ad network is charged. Since the ad exchange requires real-time bidding from the ad networks, what strategy should be used to maximize their profits becomes an emergent question.

The Trading Agent Competition Ad Exchange (TAC AdX) is an opensource platform for simulating ad exchange marketplace, and reflects the challenges faced by ad networks. Autonomous ad networks are able to test their algorithms and to compete on the platform without the risk of real monetary loss. As a broker, the ad network also needs to organize contracts to attract advertisers. In TAC AdX'14, up to 8 self-interested agents representing ad networks compete with each other. The objective of the team is to maximize the profit while maintaining a long-term reputation.

However, designing a highly competitive agent is not a trivial job. The challenges of designing an agent are summarized as follows.

- Each agent is required to estimate/predict the market's trend, and to submit bids with limited incomplete information of the market. Decision making for profit maximization based on partial information is always a hard problem.

- Each agent must make the trade-off between cost and long-term reputation. With the fluctuation of the market, fulfilling an ad contract within limited time may result in monetary loss at the agent, while breach of contact will lower the agent's reputation, which fur-

†F. Wu is the corresponding author.

*This work was supported in part by the State Key Development Program for Basic Research of China (973 project 2014CB340303), in part by China NSF grant 61422208, 61472252, 61272443 and 61133006, in part by CCF-Intel Young Faculty Researcher Program and CCF-Tencent Open Fund, in part by the Scientific Research Foundation for the Returned Overseas Chinese Scholars, and in part by Jiangsu Future Network Research Project No. BY2013095-1-10. The opinions, findings, conclusions, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies or the government.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

[1]In this paper, we consider the case that advertisers participant in the auction for ad slots through ad networks. Ad networks collect their subscribers' bidding strategy beforehand, and interact with the ad exchange during the ad auction.

ther lower the agent's competitiveness in future competitions.

- Each round of game lasts for a short time and produces little data. Since agents (or at least ANL) are not supposed to make use of historical data from previous rounds, zero-knowledge algorithms are required.

Our agent, named *ANL*, won the first TAC AdX competition, with a total profit 1.592 times that of the runner up. In this paper, we present detailed design of our agent *ANL*. We further design a more adaptive and active agent, namely *AdvANL*, which is even superior to *ANL* in our simulated competition environment.

The rest of the paper is organized as follows. In Section 2, we describe key elements and the flow of TAC AdX. In Section 3 and Section 4, we present the design of *ANL* and *AdvANL*, respectively. In Section 5, we show the result of TAC AdX'14, and compare *ANL* with competitors. In Section 6, the paper presents comparison results between *AdvANL* and *ANL* in controlled simulations. In Section 7, we briefly introduce related works. In Section 8, we conclude the paper.

2. TAC ADX

TAC AdX is a virtual marketplace where agents, representing ad networks, can organize marketing contracts, bid for ad impressions, fulfill the contract and get revenue. We begin with a general description of key elements in the game. Agents are required to implement software agents performing bidding strategies in auctions, while the platform simulates behaviors of Internet users, web publishers, advertisers and an ad exchange. Figure 1 [12] shows the game structure.

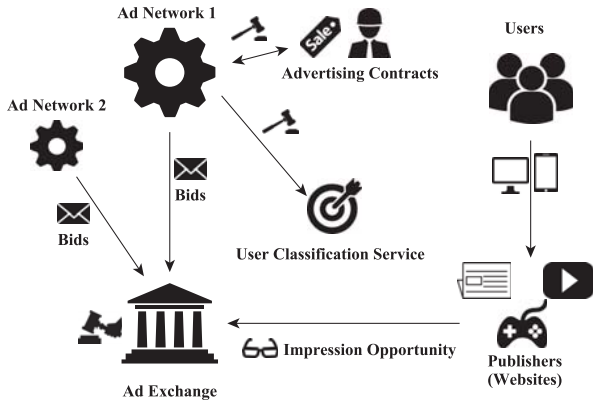

Figure 1: The structure of TAC AdX'14

Advertisers create marketing contracts with the objective of maximizing the number of impressions towards audience with certain attributes. For example, a server re-seller, as an advertiser, might be interested in people who are about to build their web sites, so the re-seller comes to the ad exchange and makes an offer: her advertisement should be displayed for at least $100,000$ times to these people, and \$0.001 is paid to the exchange for every impression.

Each contract is allocated to an agent by conducting a variant of sealed second-price auction. Once winning a contract, the agent is required to carry out the contract by bidding for impression opportunities in the ad exchange. Impression opportunity comes whenever a user visits a publisher's site, and is sold to the agent with the highest bid.

The balance, quality score and winning bid are all private to each agent, that is, if an agent does not win an auction, it has no idea about who wins or what the winning bid is. For more information, please refer to the game specification [12].

The game proceeds in a series of simulated days, typically 60. During each simulated day, each agent is asked to make several decisions in markets. Each game takes about 10 minutes in the simulator. Agents act as ad networks and are evaluated by their net accumulated bank account balances over all games. The agent starts with \$0.0 balance at the beginning of each game and is allowed to overdraw.

Agents compete with each other by taking actions in three auctions during each simulated day:

(1) one marketing **contract auction**, where a new advertising contract is announced, and auctioned. The auction outcome is released at the start of the next day;

(2) one **User Classification Service (UCS) auction**, which determines the agent's abilities to access user attributes;

(3) a bundle of **impression auctions**, where agents bid for ad impressions to fulfill their ongoing contracts.

Figure 2 shows the game flow[2] of AdX in each simulated day, together with the design of *ANL* at high level.

The three auctions are described as follows.

2.1 Contract Auction

TAC Ad Exchange partitions Internet audience into disjoint segments. All these segments form a collection $\mathcal{S}$. Each segment $s \in \mathcal{S}$ represents a combination of user attributes. For example, a user may be described as $s =< MALE \wedge YOUNG \wedge LOW\_INCOME >$. Population of a segment s is prior knowledge, denoted as $W(s)$, and similarly, that of a collection of segments S is denoted as $W(S)$.

Every day, an incoming marketing contract opportunity $C_i(T_i, S_i, R_i)$ is broadcast to agents, where T_i is a series of days the contract is conducted in, $S_i = \{s_{i1}, s_{i2}, ...\}$ is a set of targeted user segments, R_i is the number of ad impressions *at least* to reach among targeted users. Each agent then submits its bid price $b_{k,i}$ for the contract. With probability 36%, defined in the game specification, the contract is randomly assigned to an agent exactly with its bid price as the budget, otherwise the contract is allocated to the agent k with the highest effective bid $e_k = Q_k/b_{k,i}$, where $Q_k \in (0, 1.385)$, named *quality score*, is decided by the percentage of completion of its previous contracts, which intuitively represents the reputation of the agent. The budget $B_i = Q_{win}/b_{second}$, where b_{second} is the bid price of the agent with the second highest effective bid. Each agent's bid should be within an interval with respect to Q_k, indicating that a higher Q_k leads to a wider range for candidate budget. Agents with poor Q_k , i.e., $Q_k < 0.316$ are eliminated from the game. The next day, the winner of the previous contract is announced to all agents. However only the winner is notified of the actual budget of B_i. No one but the winner can tell if the contract is allocated randomly or not.

[2]There is minor difference between the flow described in the paper and that in the game specification without loss of accuracy.

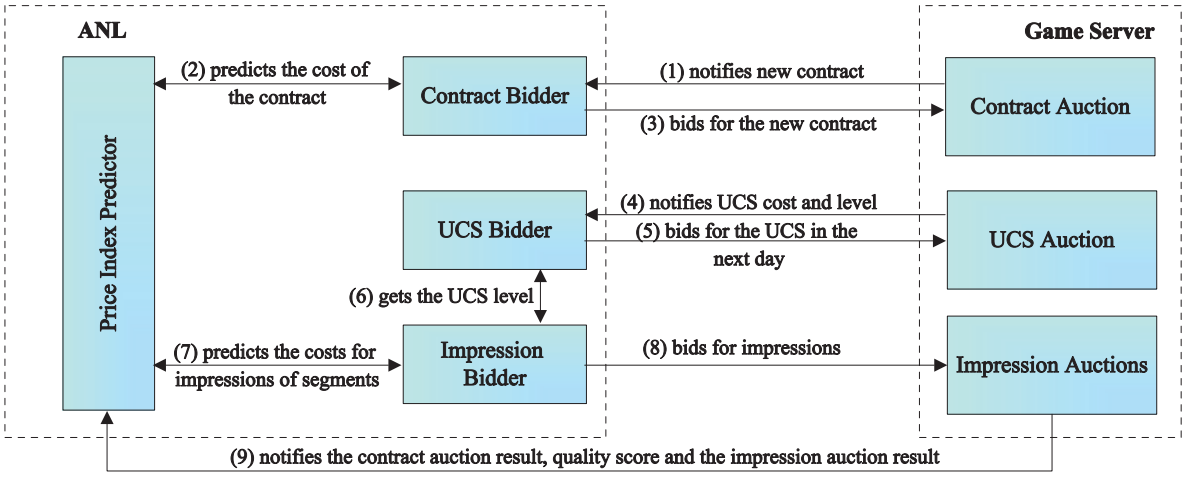

Figure 2: Game flow and *ANL* modules

Once the contract meets its deadline, the effective reach ratio is calculated by function $q(\eta)$[3], where η is the percentage of completion of the contract, e.g., $q(100\%) = 1.0$. Overfulfilling ($\eta > 1.0$) results in $q(\eta) > 1.0$. The agent is paid $q(\eta)B_i$, and Q_k is immediately updated as $0.6Q_k + 0.4q(\eta)$.

Introducing stochastic allocation protects agents from being always dominated, otherwise agents with higher Q_k would dominate the game. Incomplete auction result obscures agents about the marketing conditions. How to make decision will be discussed in Section 3.

2.2 User Classification Service Auction

In TAC AdX, for any incoming impression opportunity, each agent can reveal the related user attributes by chance. Each agent bids impressions from certain user segment, and the user segment is revealed correctly with a probability depends on the agent's current UCS level. The UCS level and the cost are determined by a Generalized Second-Price (GSP) auction on the last day. The auction is conducted daily and the result is sent to agents separately.

We discuss details of the UCS auction in Section 3.4.

2.3 Impression Auction

Impressions are bid in a sealed second-price auction with reserve price. A reserve price is set for each impression opportunity based on a segment specific property, namely *baseline*. The reserve price is randomly generated following a uniform distribution in an interval of fixed length 2Δ whose mean is *baseline*. During each day the *baseline* of each segment remains unchanged, and at the end of a day, *baseline* is updated in the direction of the reserve price resulting in the highest average profit through the day. And if there is no bidding for a certain segment at all during the day, the *baseline* drops by 40%.

Intuitively, bidding high price results in *baseline* rising. Some properties are discussed in Section 4.2.

3. THE CHAMPION AGENT

In this section, we formulate the problem, and describe modules of the champion agent, namely *ANL*, in detail.

[3] $q(\eta) = 0.4895(\arctan(4.08577\eta - 3.08577) + 1.2574)$. Thus $q(1.0) = 1.0$.

3.1 Problem Formulation

To maximize the utility, our agent ANL focuses on optimizing long-term contract income, UCS cost and impression cost while maintaining Q in a proper range. (In the following sections, we do not distinguish Q and Q_n.) Denote the contract auction outcome indicator as

$$\varphi_i = \begin{cases} 1 & \text{if } C_i \text{ is assigned to ANL} \\ 0 & \text{otherwise.} \end{cases}$$

Let $U_i(B_i, R_i, I_i, p_i) = (B_i q(I_i/R_i) - p_i I_i)\varphi_i$ be the profit gained from contract C_i, where I_i is the impressions achieved, p_i is the average cost for each impression.

Two symbols are introduced here:

Price Index (PI) indicates the estimated impression cost of segments in next one or several days.

Competing Index (CI) describes how desperate ANL is to win a contract. Higher *CI* results in lower bid price, and typically a better chance to win a contract auction.

Since the information of contracts is unknown in advance, it is computationally intractable to maximize the total profit. Thus we consider to maximize the profit for each contract by assuming $\varphi_i = 1$:

$$\arg\max_{\eta, b_i} \quad E[U_i] = b_i q(\eta_i) - PI \cdot \eta_i R_i \tag{1}$$

$$\text{subject to} \quad \frac{R_i}{Q} < b_i < \frac{R_i Q}{10} \tag{2}$$

$$b_i \le R_i \cdot PI \cdot CI \tag{3}$$

$$0.9 \le \eta_i \le 1.3, \tag{4}$$

where $b_i = b_{ANL,i}$ is the bid price from *ANL*. (2) is defined by the game specification. (3) implies the estimated private value. (4) indicates the upper and lower bounds of the percentage of completion, which results in quality score Q in a reasonable range.

To maximize (1), we pick the largest integer b_i constrained by (2) and (3). Then (1) can be reduced to an equation in the form of $X\arctan(\eta_i) - Y\eta_i + Z$, which can be easily solved, where X, Y, Z are constants.

Therefore, we are able to make our best response every day based on estimated *competing index* and *price index*.

3.2 ANL Overview

We first present a high-level overview of *ANL*. Generally speaking, *ANL* is characterized as "*active, generous and gamble*".

Active: ANL is always eager to win a contract if the contract is evaluated as possible to complete. Failing to win a contract leads to ANL expecting less profit.

Generous: ANL bids for impressions with high budget, in order to maintain a considerable Q.

Gamble: if an incoming contract is estimated with a too high *price index*, *ANL* regards it as *too difficult* to complete, and bids as high as possible waiting for a stochastic assignment.

During each simulated day, ANL takes steps described in Figure 2. To make it clear, we briefly explain some of these steps.

Step (2): Contract Bidder offers Price Index Predictor the information about the incoming contract, including audience segments and minimum reaching impressions. Price Index Predictor estimates whether there is vacancy in segments in the next few days, calculates the *price index* of each segment and sends to Contract Bidder.

Step (5): to receive a considerable UCS level in the next day, UCS Bidder determines the bid price by referring to the previous bid price and the UCS level received in Step (4).

Step (7): Impression Bidder looks up for all owned contracts, collects related segments, and submits them to Price Index Predictor. Predictor estimates PI of each segment for the incoming impression auctions, and sends them to Impression Bidder.

In ANL, two parameters are required to set.

- $G_{greed} > 1$ is a factor describing how greedy ANL is while bidding for a contract. It is used to update the competing index. ANL is more generous if G_{greed} is set high.

- $G_{UCS} \in (0.5, 1)$ means the percentage the bid in UCS auctions should be scaled up by in order to receive a higher UCS level. It helps to determine the critical value in UCS auctions, that is, the upper bound of ANL's bid in UCS auctions. Meanwhile, ANL changes its bid in UCS auctions with respect to G_{UCS}.

Neither G_{greed} nor G_{UCS} requires accurate estimation, since ANL regards them as reference only, and they never directly affect the cost. To let ANL be generous, we set $G_{greed} = 1.2$ and $G_{UCS} = 0.2$ in the competition. In fact, any arbitary values would make little difference.

ANL does not consider the bidding strategy of each competitor, because it treats opponents' bids as a part of the whole distribution of the bids in the market.

3.3 Contract Bidder

The marketing contract is allocated in a mixed-second-price sealed auction, where the contract is stochastically assigned to an agent with probability 0.36. Under this circumstance, the auction is not truthful, and bidding the private value may not be optimal for agents.

We summarize strategies into three main categories:

(1) bidding the private value, which is derived from the estimated *price index* denoted as PI, and *competing index* denoted as CI;

(2) bidding the lowest valid price, if the quality score of the agent is relatively low;

(3) bidding the highest valid price.

The *price index* is a segment indicator to estimate the cost in the impression market, and the *competing index* determines the expected return on investment. Both of them jointly decide the private value,

$$PrivateValue = CI \cdot PI.$$

In this market, the quality score Q positively determines the lower and upper bound of the bid price. Same bid price with a higher Q results in both a higher effective bid and possibly a higher budget. Therefore, maintaining a considerably high Q is quite necessary in the long run. Once Q is lower than the expected value (0.8), *ANL* follows Strategy 2 to desperately to win a contract in order to raise Q. Strategy 3 is applied whenever an incoming contract is evaluated as too difficult to fulfill. *ANL* regards a contract as too difficult, if the average *price index* of the related segments is too high.

At a high level, whenever *ANL* fails to win a contract, it raises the *competing index* to decrease the bid price and expects higher effective bid. Once ANL wins a contract, it is able to figure out whether the contract is allocated randomly. *ANL* lowers the *competing index* if the auction is a standard sealed second-price auction. If the contract is randomly assigned to *ANL*, the *competing index* remains unchanged.

To summarize, the new competing index CI_{new} is updated as follows,

$$CI_{new} = \begin{cases} CI & \text{if ANL wins, budget = bid} \\ CI/G_{greed} & \text{if ANL wins, budget} \neq \text{bid} \\ G_{greed} \cdot CI & \text{if ANL fails to win.} \end{cases}$$

3.4 UCS Bidder

UCS levels and costs are determined in a generalized second-price sealed auction. In the auction, the agent with the k^{th} highest bid receives the UCS level of $l = (0.9)^{k-1}$, and pays the amount of the $k+1^{th}$ agent's bid with a discount $1-l$, that is the candidate UCS levels are $1.0, 0.9, 0.81, ...$ An agent with UCS level 0.9 will fail to reveal attributes of users with probability 0.1, which means that it cannot bid in 10% impression auctions.

We presume that the impression unit-price

$$p \propto demand/supply,$$

where *demand* (*supply*) infers the demand (supply) of impressions in certain segments. Therefore the UCS level decreasing by 10% is equivalent to lowering the supply by 10%.

Intuitively, the higher the UCS level is, the more impressions an agent is able to reach. However, it might happen that one or more competitors competes too aggressively to receive the highest level, and as a result, the UCS cost becomes so high that the return on investment goes too low. Instead of chasing for the highest, ANL only expects some lower levels. In the extremity where costs of all levels in preference are unacceptable, ANL lowers the expectation in UCS auctions and refuses to bid higher.

Most of the time, ANL maintains the UCS level between 0.81 and 0.9 (i.e., the 2^{nd} or the 3^{rd} highest) to avoid cut-throat competition and to prevent from too many off target

impressions. However if the cost to maintain the level is too high, the agent will stop raising the bid to follow the level.

Assume at day t, ANL has r (uniformly defined in $[r_0, 2r_0]$) impressions to go, where r_0 is the minimum impressions the agent will reach tomorrow, and $2r_0$ is the maximum, respectively. Suppose the current bid for UCS is β, and a G_{UCS} rise of β tends to increase the UCS level by 0.9 times. We can estimate the critical value with the following equations:

$$\begin{aligned}
&E[UtilityIncrement] \\
=&\int_{r_0}^{2r_0} (DecreasedImpCost - ExtraUCSCost)\frac{1}{r_0}\mathrm{d}r \\
=&\frac{1}{r_0}\int_{r_0}^{2r_0} r\left(p - p'\right) - (1+G_{UCS})\beta \mathrm{d}r \\
=&\frac{1}{r_0}\int_{r_0}^{2r_0} \left((1-0.9)\, rE[p] - (1+G_{UCS})\,\beta\right)\mathrm{d}r = 0,
\end{aligned}$$

where p' is the average impression price as the UCS rises by one level, that is, $p' = 0.9p$ according to the specification.

Solve the equation above, we have

$$\frac{r_0}{\beta} = \frac{20}{3}\cdot\frac{1+G_{UCS}}{E[p]}.$$

Thus, ANL will never raise the bid price once $\frac{r_0}{\beta} < \frac{20}{3}\cdot\frac{1+G_{UCS}}{E[p]}$, where $E[p]$ is calculable according the game specification. In practice, r_0 can be estimated with respect to daily reach, that is, the expected impressions to go in the next simulated day.

Algorithm 1 shows the pseudo-code of UCS bidding strategy, where *DailyReach*() is calculated assuming that at every point of time an equal share of contracts is completed. ANL sets $r_0 = 3/4\mathit{DailyReach}()$.

Algorithm 1: Determine UCS Bid

Input : The previous bid price β, received UCS level l
Output: Today's bidding price

```
r_0 ← (3/4) · DailyReach();
if l > 0.9 then
    return β/(1 + G_UCS);
else if l < 0.81 and r_0/β >= (20/3) · (1+G_UCS)/E[p] then
    return (1 + G_UCS)β;
else
    return β;
```

3.5 Price Index Predictor

As described in Figure 2, Price Index Predictor is one of the core modules in ANL. It interacts with most of modules and handles two types of requests.

(1) **One-day predicting request**: estimating the *price index* of a given segment for the next day.

(2) **Multi-day predicting request**: estimating the average *price index* of a collection of segments for the following days.

The former one is heavily used by Impression Bidder for preparing bid bundles and the latter one mostly helps Contract Bidder to determine contract bids.

Contracts are processed by buying impressions in the impression market, and agents get paid according to the percentage of completion of contracts. In reality, whenever a user opens a web page with an ad slot, an impression auction is conducted right away, and ad networks who are interested in the slot submit their bids in real time. In AdX, however, it is not feasible to run such large amount (approximately 10,000 times) of auctions in each simulated day.

Instead of online bidding, agents are required to submit a bundle of bids in advance every day. A bundle is contract specific, and contains both the daily cost limit and daily impression count limit. Each bid in the bundle should contain the following information: targeted segment, impression type (combination of pc, mobile and text, video), bid price and the weight of such bid. Whenever a user (impression) comes, the platform conducts an auction internally by

(1) determining the segment and the type the impression is in, and generating a reserve price around *baseline*, (2) filtering bids targeting such segment and type, and deciding at most one bid from each agent by playing roulette, (3) running a second-price auction to determine the winner and the payment.

At the end of each day, the *baseline* of each segment is updated. Generally speaking, for a popular segment, the higher the bid price is, the higher the *baseline* will be. On the other hand, if a segment is not hot at all, the *baseline* drops.

PROPERTY 1. *In the impression market, the baseline of a segment rises if the segment is popular, and drops otherwise.*

Taking Property 1 into consideration, we understand that estimating the *price index* accurately is essential to make decisions and the key to winning the game.

In the competition, *ANL* directly sets *price index* of each segment s in day t as its popularity $pop(s,t)$. Assuming the contract is processed with even rate, the popularity of an atomic segment s in Day t is defined as,

$$pop(s,t) = \sum_{s\in S_i \wedge t\in T_i} \frac{R_i}{W(S_i)|T_i|}. \tag{5}$$

Thus we can define the popularity of a collection of segments S during a set of days T as

$$pop(S,T) = \frac{\sum_{s\in S, t\in T} W(s)\cdot pop(s,t)}{|T|\cdot W(S)}. \tag{6}$$

3.6 Impression Bidder

Impression Bidder is in charge of determining bid bundles to fulfill contracts assigned to ANL. It basically sets bid price with respect to the *price index* from Price Index Predictor. However, there does exist a challenge: potential budget deficit problem.

As a place with autonomous agents, the impression market changes sharply. As the market evolves, chances are that the price of impression rises, and the cost for impressions is beyond the budget, meanwhile, the contract is not completed yet. In this condition, the loss of money directly effects the profit (score), while the loss of quality score leads to the agent more vulnerable in the rest of the game. Therefore, the agent has to make trade-offs between the money and the quality score.

Considering losing profit is a one time loss, and losing the quality score Q constantly affects the agent's competitive power, we regard maintaining Q as a priority task. To

balance the profit and the quality score, ANL spends the budget obeying the following rules.

(1) ANL always sets the cost limit as a fixed fraction of the remaining budget.

(2) ANL raises the bid price if the deadline is approaching and the contract is not completed.

ANL submits a bundle of bids for each ongoing contract, with the unit-price of impressions calculated by Algorithm 2, where $R_{ContractMax}$ is defined in the specification.

Algorithm 2: Decide Impression Price

Input : Impressions to go *reach*, remaining budget *budget*, the percentage of completion so far η
Output: Bid price for each impression

```
budget ← fixed percentage of budget;
if Too little budget with small reach then
    budget ← R_ContractMax · budget;
if Only one day left and η < η_i then
    Double the budget;
return budget/reach;
```

At a high level, *ANL* treats Q more important than the profit. Whenever a contract comes to the end with a low completeness, budget is immediately doubled. *ANL* fulfills contracts with only a few impressions to go with a high unit-price, and this results in a high Q but small total cost.

4. ADVANCED AGENT

In this section, we introduce a more adaptive and active agent, called *AdvANL*.

4.1 Responsive Price Index Estimation

From the competition result, we find that, the impression bid price *ANL* submitted is much higher than the purchase price (8.38 times in average). In Section 2.3, we drew the conclusion that bidding impressions with high price causes *baseline* rising, and as a result, impression prices keep going up in proceeding days.

In this section, we propose a novel *price index* estimation algorithm with learning process, namely Responsive Price Index Estimation (RPIE).

ANL assumes that every contract is processed with the objective of reaching 1.0 percentage of completion. In *AdvANL*, we suppose that the estimated percentage of completion of contracts $\hat{\eta}$ is a random variable following $N(1.0, 0.0044)$, i.e., $\hat{\eta} \in [0.8, 1.2]$ accounts for 99.7%. Equation (5) is updated as

$$pop_{adv}(s,t) = \sum_{s \in S_i \wedge t \in T_i} \frac{\hat{\eta} R_i}{W(S_i)|T_i|}. \quad (7)$$

Besides, $pop_{adv}(S,T)$ is defined similar to Equation (6).

The main assumption supporting RPIE is that *price indexes* of different segments with similar *popularity* obey the same distribution in the same periods of time, while the relation between *price index* and *popularity* evolves gradually. RPIE stores the daily impression bidding results as knowledge. As time goes by, the weighting for each older data decreases exponentially in the procedure of estimating *price index*. Every day, *AdvANL* receives the impression bidding report from the game server, containing a list of tuples in form of (C_i, *ReachedImpressions*, *TotalPayment*). We define the *price index* of segments S_i at the last day t as $\rho = TotalPayment/ReachedImpressions$. Denote δ as the *popularity* of S_i at day t, thus with Equation (7), we have $\delta = pop_{adv}(S_i, \{t\})$. RPIE collects all (δ, ρ, t) tuples as labeled examples.

A KNN-like regression algorithm is used to estimate the *price index*. Given two tuples $(\delta_1, \rho_1, t_1), (\delta_2, \rho_2, t_2)(t_1 \le t_2)$, the distance is calculated as follows

$$Distance = |\delta_2 - \delta_1| \cdot \alpha^{t_2 - t_1},$$

where α is the attenuation coefficient. Since the contract lasts for at most 10 days, we expect weighting of the tuple added 10 days ago decreases to 0.1 or less. Thus we let $\alpha = \sqrt[10]{0.1} = 0.794$.

Given the *popularity* δ and the day t, RPIE picks K labeled examples running the classic KNN algorithm, denoted $(\delta_i, \rho_i, t_i), i = 1, 2, \ldots, K$. Performing the weighted linear least squares algorithm with the K labeled examples, we obtain a linear function with which *price index* ρ can be calculated. Algorithm 3 shows RPIE in detail.

Algorithm 3: Responsive Price Index Estimation

Input : A list of labeled examples e, the day t, a set of segments S
Output: Estimated *price index*

```
δ ← pop_adv(S, {t});
Sort labeled examples e by increasing distance;
if |e| < K then
    return Random a price;
else
    for i from 1 to K do
        x[i] ← e[i].δ;
        y[i] ← e[i].ρ;
        w[i] ← α^(t − e[i].t);
    f ← WeightedLinearRegression(x, y, w);
    return f(ρ);
```

4.2 Impression Bidding

We first discuss the two properties of the impression market.

As mentioned in Property 1, the *baseline* of an unpopular segment drops exponentially. This property implies that the *price index* of a segment with low cumulated popularity should remain low.

From the game specification, the reserve price for each impression is between $\Delta_{low} = baseline - \Delta$ and $\Delta_{high} = baseline + \Delta$, where $\Delta = 0.0001$. Since the reserve price is uniformly randomized, the probability of an impression bid $x \in [\Delta_{low}, \Delta_{high}]$ not being banned from any auctions is $(x - \Delta_{low})/2\Delta$, which indicates that the supply of such segment squeezes. Therefore, we have the following property.

PROPERTY 2. *In the impression market, for a specific segment, the bid x is able to attend all auctions only if $x > \Delta_{high}$.*

Property 2 implies that the agent should bid the price higher than Δ_{high} if it is desperate for impressions.

AdvANL regards the *price index* estimated by RPIE as the *baseline* of each segment. Instead of bidding the same price

among all segments in a bundle, which is *ANL*'s strategy, *AdvANL* bids different prices for segments based on the corresponding estimated *price indexes*. For contracts with low percentage of completion, *AdvANL* doubles the bid price, and adds another Δ to the bid according to Property 2.

5. COMPETITION RESULTS

The TAC AdX'14 was held in conjunction with AAMAS'14 conference. In this section, we present competition results.

There are 8 competitors involved in the 3-day competition. In each day, 40 rounds of games were held, and each round consists of 60 simulated days. Agent ANL never competed with other agents until the competition started, and ANL was never modified during the competition.

5.1 Result Overview

Table 1 illustrates the cumulative scores in each day. As is shown, ANL achieved the highest score in both Day 1 and Day 3, and behind *giza* in Day 2 with gap of only $44.6. The total profit (score) achieved by *ANL* is 1.592 times that of the runner up.

We analyze these results in the following sections.

Table 1: Results ($) of the TAC AdX'14

Agent	Day 1	Day 2	Day 3	Total
ANL	1098.4	704.0	1266.5	3068.9
giza	447.8	748.6	732.4	1928.8
Agent2	596.8	454.5	363.6	1414.8
tau	-93.8	488.0	727.7	1122.0
livadx	233.5	227.7	123.4	584.5
blue	60.1	169.8	82.3	312.2
WinnieTheBot	-0.8	119.5	80.6	199.4
Amunra	-54.0	-1.8	0.0	-55.9

5.2 Active and Inactive Agents

Over the competition, four agents outperformed the others, which are *ANL*, *giza*, *tau* and *Agent2*, and we call them active agents. Meanwhile, the other four, namely *livadx*, *blue*, *WinnieTheBot* and *Amunra*, achieving the profits not as good as active ones are called inactive agents.

As mentioned above, in the contract market, only agents with valid Q are allowed to bid. From game log files, we notice that some agents (*livadx* and *blue*) stayed with $Q < 0.4$ for more than 50% of the game, which almost wiped them out of the contract market for quite a few of simulated days. We formally regard an agent as **active** if the percentage of time that its Q under the bottom line(0.4) throughout the games is no more than 20%.

On the one hand, as discussed in previous sections, this game itself is complicated: agents are required to make several decisions in each round, and each choice made by an agent might lead to the butterfly effect. Therefore we believe that it is very difficult to make decisions even if the game involves only 3 competitive participants. On the other hand, each action taken by an agent makes negative influence on all others, that is, it is not surprising to see some agents are suppressed by others.

After looking at logs, we believe that at least 7 agents did a good job during the competition, though three of which did not do a great job. These 3 agents are vulnerable and are most likely be suppressed by the active ones, considering that the top four are all very aggressive and competitive.

Besides, it is the poor performance of the bottom four agents that convinces us that ANL did well in both suppressing opponents and adjusting to the environment in time.

5.3 Quality Score Maintaining

ANL achieved a great success in maintaining the quality score Q throughout the competition. Figure 3 shows the four active agents' cumulated distribution of quality score. The quality scores of *giza*, *Agent2* and *tau* change similarly and are almost uniformly distributed from 0.0 to 1.4.

Meanwhile, ANL maintains a relatively high quality score for most of the time. Specifically, its quality score is higher than 0.9 and 1.2 for 92.41% and 74.32% of the time respectively, while below 0.54 for only 2.04% of the time.

The high quality score enables *ANL* to achieve higher effective bids than others and to become more competitive with the same bid price in the contract market.

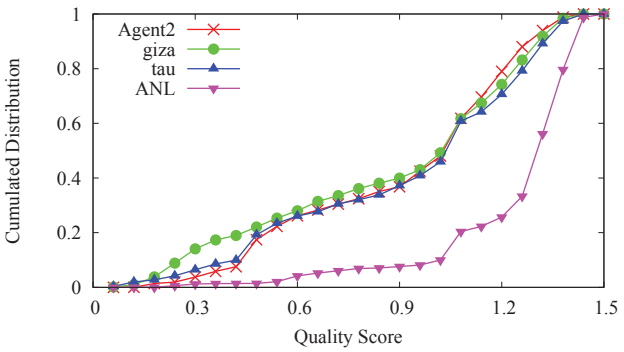

Figure 3: Cumulated distribution of quality score. ANL maintains the quality score above 1.0 for exactly 90.02% time in the competition.

5.4 Profiting Efficiency

Figure 4 demonstrates the accumulation of primary cash-related components during the competition.

Intuitively, *giza* and *Agent2* won impressions with a much lower cost (roughly 50% lower than *ANL*), but did not fully complete contracts. *giza* and *Agent2* reached the percentage of completion 89.9% and 87.7%, respectively, which means that both of them strictly limited their budget in the impression market even with some loss of quality score.

From the figure, it is clear to see that ANL is more efficient in making profit.

The budgets of *ANL* ($4941.1) and *giza* ($4567.6) are very close. And the costs of the two are relatively comparable. Assuming that both of *ANL* and *giza* achieved 1.0 percentage of completion in average, the expected profits should be 2396.7 and 2837.3, respectively. However, *ANL* gains $1087.3 more profit than *giza*. The average percentage of completion of *ANL*'s contracts is 1.27, while that of giza's contracts is only 0.89, which implies that *ANL* earns 48.18% more profit at the cost of only 0.38 percentage of completion.

6. CONTROLLED SIMULATIONS

In this section, we conduct controlled simulations to compare the performance between *AdvANL* and *ANL*.

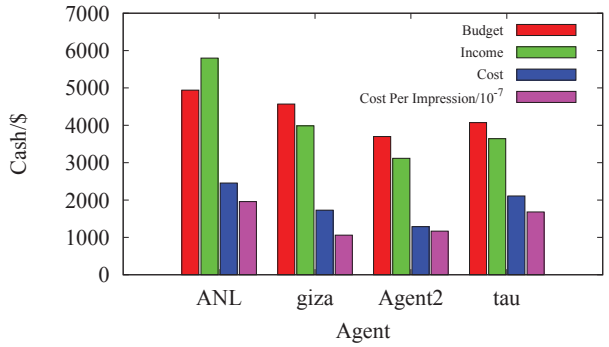

Figure 4: Accumulated contract budget, actual income, impression cost and impression unit-price of the top four agents in the competition

We run simulations with the raw data from TAC AdX'14, and each agent (original agent) behaves exactly the same as it did in the competition. Agents behaving independently according to the data are called live agents. In the simulation, only live agents are compared.

6.1 Profit Comparasion

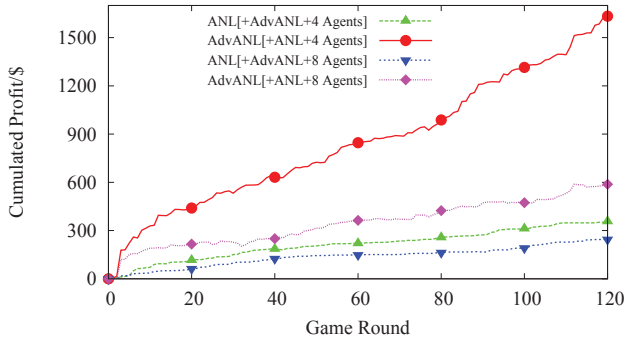

Figure 5: Cumulated profit of live AdvANL and live ANL though 120 rounds of simulations. Content in brackets infers accompanied agents. "4 Agents" indicates the top four agents.

Figure 5 illustrates the cumulated profit of live *AdvANL* and *ANL* with different number of original agents, who played in the competition. "4 Agents" indicates the original four most active agents, i.e., *ANL*, *giza*, *Agent2* and *tau*.

In "*ANL* + *AdvANL* + 4 Agents" scenario, live *AdvANL*'s profit increases steadily and reaches $1633 eventually, whereas live *ANL*'s profit remains as low as $356 at the end of the simulation, which is approximately 15.7% of that of *AdvANL*. In the game involving live *AdvANL*, live *ANL* and 8 original agents, live *AdvANL* again outperforms live *ANL*. Live *AdvANL* achieves this advantage because of its higher activity and fine-grained market prediction.

It is interesting to see that the profit achieved by AdvANL with 9 competitors is much less than that made by AdvANL with only 5 opponents. It implies that any action taken by agents suppresses others, and the 4 inactive agents did not perform poorly at all.

6.2 RPIE Performance

The major difference between AdvANL and ANL is that RPIE is applied by AdvANL. As discussed in Section 4.2, higher ratio leads to impression reserved price rising. And as a result, the cost increases in the following days.

Figure 6 plots the ratio between bid price and purchase price in impression auctions throughout 120 rounds of games. *ANL* achieves the ratio 8.38 in average, while that of *AdvANL* is 4.19. Meanwhile, the standard deviation of the ratio from AdvANL is 1.40, and that from ANL is 4.52, which indicates that RPIE performs well and is considerably stable.

With the help of RPIE, *AdvANL* estimates the *price index* with acceptable accuracy, and avoids unnecessary rises in impression price.

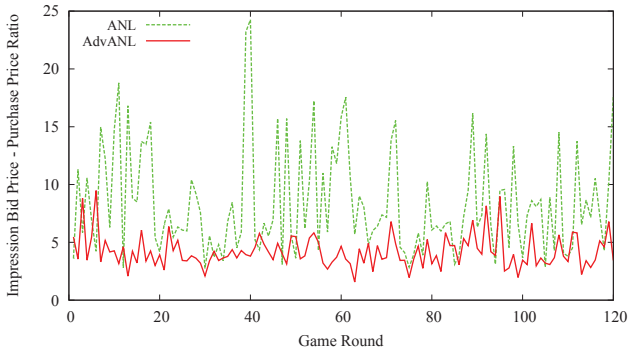

Figure 6: The ratio between bid price and purchase price in the impression market throughout 120 rounds of games.

7. RELATED WORK

While ad exchanges have drawn considerable attention by both companies and researchers in recent years, the majority of related publications only focus on mechanism design and allocation.

Since TAC AdX'14 is the first TAC AdX competition, there is no published work on ad network agent design. The closest game TAC Ad Auction [11] (TAC/AA), which is an online advertising competition, is built without reputation system, and contracts are assigned to agents by the server actively. Chang et al. [5] presents a feasible market predicting method in TAC Ad Auction based on analyzing key elements of the game.

Urieli and Stone [13] introduced a Markov Decision Process (MDP) model based on [6] in the wholesale market in the game Power TAC'13. However, AdX marketplace is too complex to apply MDP. TAC Supply Chain Management [10] is a game, where agents deal procurements and sales which are similar to TAC AdX. However, it does not consider the long-term reputation either. Merrill [9] designed an adaptive agent using an interdependent optimization in TAC Supply Chain Management. Altman [4] introduced a technique for estimating a regression curve without strong assumptions.

8. CONCLUSION

This paper has introduced *ANL*, the champion autonomous agent for the realtime ad exchange competition, TAC AdX'14. We have described key elements of the competition, and presented the design of the champion agent. Furthermore, we have proposed a more adaptive and active agent *AdvANL*, the performance of which is even better than that of *ANL* in controlled simulations.

REFERENCES

[1] Doubleclick. http://www.google.com/doubleclick/.

[2] Microsoft ad exchange. http://advertising.microsoft.com/exchange.

[3] Yahoo! right media. https://advertising.yahoo.com/.

[4] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[5] M. Chang, M. He, and X. Luo. Designing a successful adaptive agent for tac ad auction. In *Poceedings of European Conference on Artificial Intelligence (ECAI)*. 2009.

[6] T. Gerald and B. J. L. Strategic sequential bidding in auctions using dynamic programming. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 591–598, 2002.

[7] Interactive Advertising Bureau. Iab internet advertising revenue report, 2014.

[8] Y. Mansour, S. Muthukrishnan, and N. Nisan. Doubleclick ad exchange auction. *Computing Research Repository*, abs/1204.0535, 2012.

[9] P. D. Merrill. *Adaptive trading agent strategies using market experience*. Ph.d. dissertation, University of Taxes at Austin, 2011.

[10] S. Norman, A. Raghu, J. Eriksson, F. Niclas, and J. Sverker. Tac-03: A supply-chain trading competition. *AI magazine*, 24(1):92, 2003.

[11] J. PatrickR. and W. MichaelP. Designing the ad auctions game for the trading agent competition. In *Proceedings of IJCAI-09 Workshop on Trading Agent Design and Analysis*, 2009.

[12] M. Schain and Y. Mansour. The ad exchange game - specification. https://sites.google.com/site/gameadx/, 2014.

[13] D. Urieli and P. Stone. TacTex'13: A champion adaptive power trading agent. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence(AAAI)*, 2014.