In *Proceedings of the Tenth International Conference on Autonomous Agents and Multiagent Systems* (AAMAS 11), Taipei, Taiwan, May 2011.

A Particle Filter for Bid Estimation in Ad Auctions with Periodic Ranking Observations

David Pardoe and Peter Stone
Department of Computer Science
The University of Texas at Austin
{dpardoe, pstone}@cs.utexas.edu

ABSTRACT

Keyword auctions are becoming increasingly important in today's electronic marketplaces. One of their most challenging aspects is the limited amount of information revealed about other advertisers. In this paper, we present a particle filter that can be used to estimate the bids of other advertisers given a periodic ranking of their bids. This particle filter makes use of models of the bidding behavior of other advertisers, and so we also show how such models can be learned from past bidding data. In experiments in the Ad Auction scenario of the Trading Agent Competition, the combination of this particle filter and bidder modeling outperforms all other bid estimation methods tested.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation, Economics

Keywords

agent modeling, learning, particle filters, trading agents, sponsored search, ad auctions

1. INTRODUCTION

Sponsored search [5] is one of the most important forms of Internet advertising available to businesses today. In sponsored search, an advertiser pays to have its advertisement displayed alongside search engine results whenever a user searches for a specific keyword or set of keywords. An advertiser can thereby target only those users who might be interested in the advertiser's products. Each of the major search engines (Google, Yahoo, and Microsoft) implements sponsored search in a slightly different way, but the overall idea is the same. For each keyword, a *keyword auction* [6] is run in which advertisers bid an amount that they are willing to pay each time their ad is clicked, and the order in which the ads are displayed is determined by the ranking of the bids (and possibly other factors). Having an ad in a higher position is generally considered to be more desirable.

Running a successful keyword advertising campaign can be difficult. An advertiser must choose the keywords of interest and its bids for each one based on an understanding of customer behavior, competitors' bidding patterns, and its own advertising constraints and needs, all of which can change over time. Complicating matters is the fact that advertisers receive very limited information about the actions taken by other advertisers. In particular, advertisers do not see the bids of other advertisers. Knowing the bids of other advertisers for a specific keyword would allow an advertiser to predict the ad position and cost per click for any amount it bid and use this information to choose the bid it expected to maximize profit. Search engines typically release some information concerning the position that an advertiser could expect for certain bids, but this information is generally incomplete and out of date. Alternately, an advertiser could experiment with different bids and observe the resulting positions, but such experimentation would be time consuming and costly.

In this paper, we present a particle filtering approach to estimating the bids of other advertisers in a single keyword auction. This particle filter relies on periodic observations of the rankings of all advertisers. In addition, it requires models of the bidding behavior of other advertisers, and we show how such models can be learned. We implement and test our particle filter in the context of the Ad Auction scenario of the Trading Agent Competition (TAC/AA) [3], a competition developed in 2009 to encourage research into keyword auction bidding within a carefully designed simulated environment. Nevertheless, the basic approach to particle filtering described here should generalize to any standard ad auction setting.

The remainder of this paper is organized as follows. After formally specifying the auction setting we consider in Section 2, we present the design of the particle filter in Section 3. Section 4 describes our experimental domain, TAC/AA, and explains how our particle filter can be applied in this domain. Our particle filter requires bid transition models for all other advertisers, and in Section 5, we present a machine learning approach to building these models. Finally, Section 6 contains experimental results comparing the accuracy of our particle filter to other bid estimation methods in three different TAC/AA settings.

2. AUCTION SETTING

We begin by formally specifying our auction setting, which

Cite as: A Particle Filter for Bid Estimation in Ad Auctions with Periodic Ranking Observations, David Pardoe and Peter Stone, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems – Innovative Applications Track (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. XXX-XXX.

has been chosen to be as general as possible while still capturing the basic elements of a keyword auction. We are interested in estimating the bids of N other advertisers for a single keyword for which we are advertising. Each advertiser has a standing bid indicating the amount it is willing to pay each time its ad is clicked, and this bid may occasionally be revised. This bid must be above a known reserve price *reserve*. When a user searches for the keyword, the advertisers' bids are ranked in descending order, and their ads are shown in this order. If more than M bids are above the reserve, then only the top M ads are shown. When a user clicks on our ad, our *cost per click* (*cpc*) is the minimum amount we could have bid and still had our ad shown in its current position. In other words, our *cpc* is equal to the amount of the bid ranked below ours, or to *reserve*. At some regular interval, we receive a report containing the following information: i) the bid ranking at time t (for advertisers whose ads are being shown, i.e., at most the top M), and ii) our own *cpc* at time t. Our goal is to estimate the bids of the other advertisers at time t. Depending on the nature of the auction, advertisers may be able to revise their bids more frequently than this reporting interval; however, we will only attempt to estimate bids at this interval, and our models of advertiser behavior will only model changes in bids at this interval (e.g., from time $t-1$ to time t).

As this model is an abstraction of the keyword auctions used in the real world, there are a number of complicating factors it does not include, but we believe it to be a useful model for study. One issue faced in real keyword auctions is that ad positions are often not determined by bid rank alone, but by a combination of bid rank and other factors such as clickthrough rate. This is not a problem, however, as in this case we could simply attempt to estimate the amount we would need to bid to achieve a higher position than each other advertiser, instead of the true bid of each advertiser, and use the same particle filtering approach.

A larger concern is that search engines do not actually provide advertisers with periodic reports of the bid rankings of other advertisers. Of course, it is possible to simply repeatedly search for the keyword and observe the order of the ads displayed, but for a large advertising campaign this process would need to be automated (using a "screen scraper"), and search engines generally take measures to prevent this type of activity. Nevertheless, a number of services offer to collect this type of information for subscribers, so the assumption of these periodic reports is not necessarily unrealistic.

Finally, we note that this auction setting is in fact an instance of a repeated generalized second price auction, and that our particle filter could be applied to any such auction given periodic ranking observations. Generalized second price auctions are most commonly used in keyword auctions, but they have been considered in other areas such as electricity auctions [10].

3. PARTICLE FILTER

We now describe our particle filter for estimating the bids of other advertisers given periodic reports. For now, we assume that we have a model of each advertiser that gives us a probability distribution over their next bid given a history of their estimated bids and rankings. Developing these models will be the subject of Section 5. Again, we emphasize that we only concern ourselves with the bids at the reporting interval — by "next bid" we mean the bid at the time of the next report, and likewise our history only reflects the auction state at the times of past reports.

Given these advertiser models and reports, we estimate the joint distribution over the bids of all other advertisers using a particle filter. A particle filter is a sequential Monte Carlo method that tracks the changing state of a system by using a set of weighted samples (called particles) to estimate a posterior density function over the possible states. The weight of each particle represents its relative probability, and particles and weights are revised each time an observation (conditioned on the current state) is received. In this case, the reports represent our observations, and each particle represents an estimate of the bids of all advertisers at the time of the last report. Additionally, each particle stores all of its past bid estimates, so each particle can be seen as a full bidding history of all advertisers. Particle filters are a fitting solution to this problem because they require no assumptions about the types of distributions involved (unlike Kalman filters), they can be used efficiently in high-dimensional spaces (unlike grid-based methods that discretize the state space), and particles are a convenient data structure for storing bidding histories. We estimate the joint distribution over bids instead of estimating each advertiser's bid independently due to the fact that our estimate for each advertiser is completely dependent on our estimate for all other advertisers. (In Section 6.2 we describe a method of estimating bids independently, but this method relies on several unrealistic simplifying assumptions.)

For the experiments of this paper, the implementation of our particle filter makes use of a discretized set of bids b^1 ... b^B, and so we describe our particle filter in terms of discrete probability distributions over these bids; however, continuous probability distributions could also be used in our particle filter if they can be dealt with analytically.

3.1 SIS Particle Filter

The simplest particle filter, and the one from which more complicated variations are derived, is the *Sequential Importance Sampling* (SIS) filter [1]. A SIS filter can be implemented for our bid estimation problem as follows. Each particle p contains a current estimate for the bids of all N advertisers, as well as bid estimates for each past time step. An initial set of particles P is chosen to reflect a possible distribution over bids when no reports have yet been received — essentially our prior. The number of particles $|P|$ should be chosen to give an acceptable tradeoff between accuracy and speed. Each particle p receives initial weight $w_p = 1/|P|$. Each time we receive a report, we update P by generating and weighting a new set of particles. For each existing particle p, we sample a new particle p' (i.e., we copy the bidding history contained in p and then sample a new set of current bids). Finally, we reweight the particles.

The sampling and weighting procedures depend on our choice of *proposal distribution* from which we sample new particles: $\pi(p'|p, report)$. π may be any distribution we choose. The weighting procedure then follows from the choice of π such that the set of weighted particles approximate the true posterior distribution. If particle p had weight w_p, then particle p' receives weight

$$w_{p'} = w_p \frac{Pr(report|p')Pr(p'|p)}{\pi(p'|p, report)} \tag{1}$$

Finally, the weights of all new particles are normalized so that they sum to one.

3.2 Choice of Proposal Distribution

The choice of proposal distribution can significantly affect the performance of the particle filter. The distribution $\pi(p'|p, report) = Pr(p'|p, report)$ is what is known as the *optimal proposal distribution* and results in a weighting of $w_{p'} = w_p Pr(report|p)$. This proposal distribution is called optimal because it results in the least variance between particle weights — $w_{p'}$ is independent of p', and so it will be the same regardless of which p' is sampled. However, the optimal proposal distribution is often not used in practice because it can be difficult to sample from this distribution and perform weight calculations. Instead, the proposal distribution that would typically be used is $\pi(p'|p, report) = Pr(p'|p)$. The resulting weighting is $w_{p'} = w_p Pr(report|p')$.

The typical proposal distribution is indeed much easier to work with in our bid estimation problem, but is has a serious flaw: $Pr(report|p')$ may frequently be zero. If too few particles receive any weight, then the filter may eventually become degenerate, with mostly identical particles. To see why $Pr(report|p')$ might be zero, recall that the report contains a ranking and our *cpc*. If the current bids represented by p' are inconsistent with this ranking, then the likelihood of p' will be zero. The fraction of inconsistent particles will depend on advertiser behavior; in the worst case of random bids, only $1/N!$ of the particles would be expected to be consistent with the ranking, as any of the $N!$ possible rankings would be equally likely. Even in less extreme cases, we would still expect there to be occasional improbable rankings. Furthermore, even if p' is consistent with the rankings, it will likely not be consistent with our observed *cpc*.

We therefore use the optimal proposal distribution in our particle filter. Particles drawn from this distribution are guaranteed to be consistent with the report. Thus, we need methods of sampling from $Pr(p'|p, report)$ and computing $Pr(report|p)$. These methods are described in the following two subsections.

3.3 Computing Pr(report | p)

For $n \in 1...N$, let a_n be the advertiser ranked nth, excluding ourselves (i.e., lower ranked advertisers have their rank increased by one). Unranked advertisers may be assigned to the remaining a values (those representing the lowest ranks) arbitrarily. For a given set of current bid estimates, let c_n indicate that a_n's bid is consistent with (i.e., not higher than) the bids of advertisers a_1 ... a_{n-1} and with our own *bid* and *cpc*. Then $Pr(report|p) = Pr(c_1 \cap c_2 \cap ... \cap c_N|p)$. That is, the probability of particle p from the previous time step leading to a new particle consistent with *report* is equal to the probability that for each other advertisers, that advertiser's new bid estimate does not exceed the bid estimate of a higher ranked advertiser or conflict with *bid* or *cpc*. Furthermore, $Pr(c_1 \cap c_2 \cap ... \cap c_N|p) = Pr(c_N|p, c_1 ... \cap c_{N-1}) \cdot ... \cdot Pr(c_2|p, c_1) Pr(c_1|p)$. Below, we show how each of these N probabilities can be computed.

For each $n \in 1 ... N$, we would like to compute $Pr(c_n|p, c_1 ... c_{n-1})$, that is, the probability that particle p leads to a new bid for advertiser a_n that is consistent with our *bid* and *cpc* and the new bids of advertisers a_1 ... a_{n-1}, *given that these bids are already known to be consistent.* This probability is computed differently for each of five different cases. Let f_n be the probability mass function for a_n's next bid given the information in p, as determined by our advertiser model for a_n. In each case, we will determine f'_n, the probability mass function for a_n's next bid given p *and* c_1 ... c_{n-1}, as well as the corresponding cumulative distribution function F'_n giving the probability that the new bid is less than (but *not* equal to, as is usual in a CDF) a given value. We begin by setting F'_0 to be 0 everywhere.

- **Case 1:** *a_n has a higher rank than us.* Because the advertiser is ranked, its bid will be consistent with the bids of a_1 ... a_{n-1} so long as its bid is no greater than the bid of a_{n-1}. Because the advertiser is ranked above us, its bid must be no less than *bid*. Therefore,

$$Pr(c_n|p, c_1...c_{n-1}) = \sum_{x=bid}^{b^B} f_n(x)[1 - F'_{n-1}(x)] \quad (2)$$

  Similarly, we can define

$$f'_n(x) = f_n(x)[1 - F'_{n-1}(x)]Z \quad (3)$$

  where f'_n has support between *bid* and b^B and Z is a normalizing constant.

- **Case 2:** *a_n is ranked one below us.* Our *cpc* is determined by the advertiser ranked below us, so we know the bid of a_n.

$$Pr(c_n|p, c_1...c_{n-1}) = f_n(cpc) \quad (4)$$

  and we define F'_n to be 0 at or below *cpc* and 1 elsewhere.

- **Case 3:** *a_n is ranked at least two below us.* As in Case 1, we need the bid of a_n to be no greater than the bid of a_{n-1}. Because the advertiser is ranked below us, its bid must be between $reserve$ and *cpc*. Therefore,

$$Pr(c_n|p, c_1...c_{n-1}) = \sum_{x=reserve}^{cpc} f_n(x)[1 - F'_{n-1}(x)] \quad (5)$$

  and

$$f'_n(x) = f_n(x)[1 - F'_{n-1}(x)]Z \quad (6)$$

  where f'_n has support between $reserve$ and *cpc.*

- **Case 4:** *a_n is unranked and there are M ranked advertisers.* Because the maximum of M advertisers were ranked, we do not know if a_n placed a bid or not. We only know that a_n's bid is no greater than the bid of a_k, where a_k is the advertiser ranked M.

$$Pr(c_n|p, c_1...c_{n-1}) = \sum_{x=0}^{cpc} f_n(x)[1 - F'_k(x)] \quad (7)$$

  and

$$f'_n(x) = f_n(x)[1 - F'_k(x)]Z \quad (8)$$

  where f'_n has support between 0 and *cpc* .

- **Case 5:** *a_n is unranked and there are fewer than M ranked advertisers.* a_n did not bid or else it would have been ranked. We treat any non-bid (or bid below the reserve) as a bid of 0, so

$$Pr(c_n|p, c_1...c_{n-1}) = F_n(0) \quad (9)$$

  and $f'_n(0) = 1$.

By proceeding through the advertisers in order, we can determine $Pr(c_n|p, c_1 \ldots c_{n-1})$ for each $n \in 1 \ldots N$ and take the product to get $Pr(report|p)$. We repeat this process for each $p \in P$ and normalize the results to obtain the distribution from which we sample when generating new particles.

3.4 Sampling from Pr(p′ | p, report)

Now given a particle p and the report, we would like to sample a new particle p'. This involves choosing a new bid b_n for each advertiser a_n, and so $Pr(p'|p, report) = Pr(b_1 \cap b_2 \cap \ldots b_N|p, report) = Pr(b_1|p, report, b_2 \ldots b_N) \cdot \ldots \cdot Pr(b_N|p, report)$.

Observe that for advertiser a_N, the function f'_N generated above is in fact the same as $Pr(b_N|p, report)$ because it represented the distribution over b_N given that the bids of all other advertisers were consistent with *report*. For any other advertiser a_n, if bids $b_{n+1} \ldots b_N$ are known, then we can compute $Pr(b_n|p, report, b_{n+1} \ldots b_N)$ by taking the highest bid of any lower ranked advertiser (if any) and normalizing the portion of f'_n above that bid. Thus, by starting with b_N and working backwards, we can sample all bids in such a way that the bids are consistent with *report* and the probability of the resulting particle p' is $Pr(p'|p, report)$.

3.5 Example

We now use an example to illustrate particle filters using both the typical and optimal proposal distributions. Suppose that there are two advertisers x and y in addition to ourselves, and that according to our advertiser models, at each time step each either increases or decrease its bid by 1, with probability 0.5 in each case. We receive a report for time $t + 1$ indicating that y had the highest bid, x had the second bid, and we had the lowest bid of 0.25. Now consider a particle p that has the following bid estimates for time t: $b_x = 2$ and $b_y = 1.5$.

For the typical proposal distribution, to sample a new particle p' reflecting time $t + 1$ we would sample new bids for each bidder according to our advertiser models. However, of the four possible outcomes, only one, $b_x = 1$ and $b_y = 2.5$, is consistent with the bid ranking. If we sampled a different set of bids for p', then the weight of p' would be set to zero.

For the optimal proposal distribution, we let $a_1 = y$ and $a_2 = x$ and follow the procedure described above. First, we determine $Pr(report|p)$. We have $f_1(0.5) = f_1(2.5) = 0.5$ and $f_2(1) = f_2(3) = 0.5$, with both functions zero elsewhere. For a_1, we follow Case 1. $Pr(c_1|p) = 1$ and $f'_1 = f_1$ because F'_0 is zero and either possible bid is above our bid of 0.25. For a_2, we follow Case 1 again. $Pr(c_2|p, c_1) = 0.25$ and $f'_2(1) = 1$, because $1 - F'_1(3) = 0$, $1 - F'_1(1) = 0.5$, and either possible bid is above our bid of 0.25. Thus $Pr(report|p) = Pr(c_2|p, c_1)Pr(c_1|p) = 0.25$, which we know is correct.

To sample a new particle p', we first sample b_2 from f'_2 and get 1, the only possibility. Then we sample b_1 from the portion of f'_1 that is above 1, and we get 2.5, again the only possibility. So we are guaranteed to sample $b_1 = 2.5$ and $b_2 = 1$, the only possibility for p' given *report*.

3.6 Resampling

This section has described an implementation of an SIS particle filter using the optimal proposal distribution. A commonly used extension of an SIS filter is the *Sampling Importance Resampling* (SIR) filter, which occasionally resamples the set of particles to prevent the weights of some particles from approaching zero. Our implemented particle filter is an SIR filter, and we resample the particles in P after each update by replacing P with $|P|$ particles sampled (with replacement) according to the weights, then setting all weights to $1/|P|$.

4. TAC/AA

We now briefly describe the experimental domain in which we test our particle filter, TAC/AA [3]. For full details, see the game specification [2]. In each TAC/AA game, eight agents compete as advertisers to see who can make the most profit from selling a limited range of home entertainment products over 60 simulated game days, each lasting 10 seconds. Products are classified by manufacturer (3) and by component (3) for a total of nine products. Search engine users, the potential customers, submit queries consisting of a manufacturer and a component, although either or both may be missing. There are thus 16 total query types. Each day, for each of the 16 query types, a keyword auction is run. For each auction, an advertiser submits i) a (real, non-negative) bid indicating the amount it is willing to pay per click, and ii) a daily spending limit (optional). The top five bidders have their ads shown in order, but if an advertiser hits its spending limit (as a result of having its ad clicked enough times), its ad is not shown for the rest of the day, and all advertisers with lower bids have their ads move up one position. Bids must exceed a small reserve price. For each query type, advertisers receive a daily report providing limited information about the results of their actions and the actions of other advertisers. Reports include the advertiser's average cpc and the average position of each other advertiser. Note that these positions, and thus an advertiser's cpc, can change throughout the day due to spending limits.

TAC/AA differs from the auction model described in Section 2 in a number of ways. First, the daily reports provide the average positions of other advertisers instead of a ranking of their bids. Fortunately, it is possible to transform average positions into bid rankings with fairly high accuracy as described in [8]. Second, in TAC/AA the reserve price is unknown, but we can obtain a reasonably accurate estimate and use this in our particle filter. Third, we are only given an average cpc. If the agent ranked one spot below us hits its spending limit before we do, the average cpc will not equal the bid of that agent, as was assumed in Case 2 above. Once again, we can use the reported average positions to determine if this was the case, and if so we can apply Case 3 instead for that advertiser. Finally, as mentioned in Section 2, in TAC/AA ad positions are determined by a combination of bid rankings and clickthrough rates. In our experiments, we address this issue by adjusting each bid of each other advertiser to be the amount we would have needed to bid to achieve a higher position than that advertiser.

The use of spending limits in general represents another significant difference. We avoid dealing with spending limits by using our particle filter to estimate each advertiser's bid at the start of the day, before spending limits cause any advertiser to drop out of the bidding. Estimating the spending limits of other advertisers can be treated as a separate problem, as in [7].

The application of our particle filter to a single query type in a TAC/AA game can therefore be summarized as follows.

On each day d, we receive a report that includes our average cpc and the average position of all advertisers on day $d-1$. We transform the average positions into the bid rankings, and we determine whether our average cpc does in fact equal the bid of the advertiser ranked below us. Then, using this information, we update our particle filter, and the result is an estimated distribution over the bids of all advertisers at the start of day $d-1$.

In our experiments, without loss of generality we consider only the nine query types in which both a manufacturer and component are specified. In any one game, a number of random factors affect the value of advertising for any particular query type, and thus the bidding behavior of the agents. The distributions from which these factors are drawn are the same for all nine query types, however, and so this bidding behavior is the same in expectation for any query type in any game. As our particle filter is designed for a single keyword auction, in our experiments we treat each game as if it provides us with nine independent and identically distributed episodes, each representing a 60-day bidding history for a single keyword.

5. ADVERTISER MODELS

In Section 3, we assumed that we had a bidding model for each advertiser so that we could determine the distribution over the advertiser's next bid given its bid history. We now describe a method of generating such a model using machine learning. While the details of this section are specific to TAC/AA, the general approach could be used in any situation in which sufficient bidding data is available for use in learning. Note that precise knowledge of bids, as is available here, is not necessary to be able to build and make use of bidder models. Estimates based on information released by search engines could be used, and it might be possible to bootstrap by alternating model building and particle filtering stages to obtain increasingly accurate estimates.

The problem we are trying to solve is a conditional density estimation problem. While a number of parametric approaches to solving these problems exist, we choose to use a nonparametric approach. The bidding behavior of advertisers can be quite complex, and we would prefer to make as few assumptions about this behavior as possible. Methods of nonparametric conditional density estimation have been used in previous TAC domains to solve problems such as predicting future hotel prices [9] and predicting the probability of an offer to a customer resulting in an order [4]. The approach we take is to learn a model that takes as input both a bid amount b and a set of features representing the current state, and outputs the probability that the advertiser's next bid is less than or equal to b. Thus by evaluating this model for different values of b, we can build the cumulative distribution function for the advertiser's next bid for any given state. This approach is similar to the one used in [9] except that rather than including a price as an input to the model, there the space of prices is discretized and the model outputs a separate probability prediction for each price.

For a given advertiser, we assume that we have access to the logs from a number of TAC/AA games in which both that advertiser and our own agent participated. From these logs, we can determine the actual bids of the advertiser as well as the reports that would have been available to our own agent at any point in time. For each day $d > 0$ and any given bid b (for which we wish to make a prediction) we generate a feature vector containing the following:

- b,
- d,
- the last five bids: b_{d-1} ... b_{d-5},
- five bid differences: $b - b_{d-1}$ ... $b - b_{d-5}$,
- the last average position: ap_{d-1},
- five average position differences: $ap_{d-1} - ap_{d-2}$ ... $ap_{d-1} - ap_{d-6}$,
- the maximum and minimum bids so far: max and min,
- the differences $b - max$ and $b - min$,
- the maximum and minimum bids over the last ten days: max_{10} and min_{10}, and
- the differences $b - max_{10}$ and $b - min_{10}$

Any reference to a day before the first day is replaced with the corresponding reference to the first day. Finally, each vector is labeled with a 1 or 0 to indicate whether the advertiser's bid on day d was less than or equal to b. We note that a five-day history is used because five-day cycles can be observed in the bid series of some advertisers (for reasons specific to the TAC/AA rules). In real auctions, a 24-hour or 7-day cycle might be more likely to occur, and an appropriate bid history could be used.

Observe that any choice of b results in a unique feature vector. To generate a set of training data from game logs, we need to choose one or more values of b to use for each bid observed. If on day d the advertiser's bid was b_d, we generate 14 training instances by using 14 different values of b. The first two values are b_d and $b_d + 0.01$. The next two values are 0 and $\hat{b}$, where $\hat{b}$ is 1.1 times the highest bid ever observed for the advertiser. Next, we divide the interval $[0, b_d]$ in fifths and choose one bid uniformly randomly from each fifth. Finally, we do the same with the interval $[b_d + 0.01, \hat{b}]$. These choices give good coverage of the range of possible bids. The number 14 was chosen to give a reasonable tradeoff between model accuracy and keeping the size of the training set manageable.

Now that we have a training set, we need to choose a learning algorithm to build our model. We experimented with the learning algorithms available in the WEKA machine learning toolkit [11] and found that M5P model trees gave the best performance both in terms of probability prediction accuracy on the data set and bid estimation accuracy of the complete particle filter. Note that our learning problem can be treated as either a regression problem (treating the probability as a number to predict) or a binary classification problem (predicting the probability of belonging to the class '1'), and so both types of algorithms were tested.

One final issue that must be dealt with is the fact that we wish to use our model to produce a cumulative distribution function, but the output of our model may not in fact satisfy the requirements. For a given state, as the bid b increases from 0 to $\hat{b}$, the output of our model should monotonically increase and reach a maximum of 1, but this will sometimes not be the case. We address this problem as follows. Let the function $g(b)$ represent the output of our model for bid b in the current state. In our particle filter, we work with

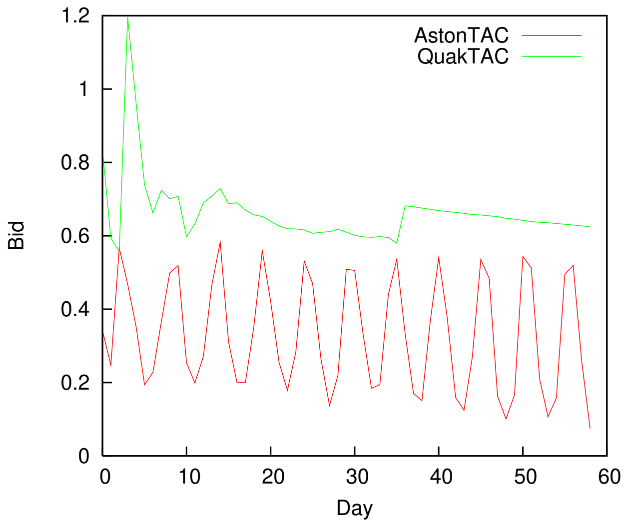

Figure 1: Daily bids of two advertisers

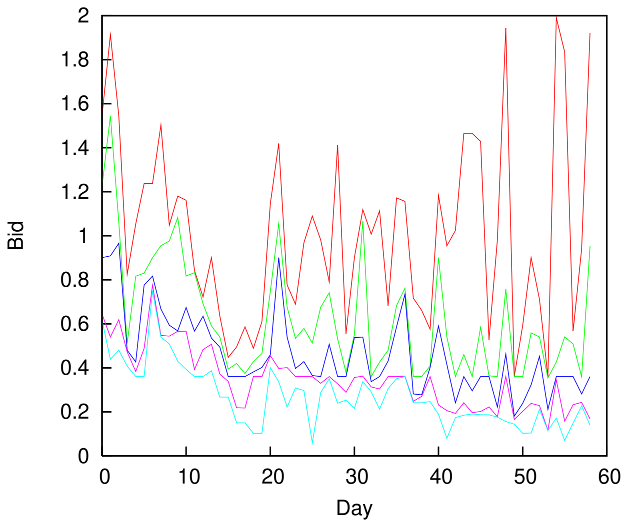

Figure 2: Top five daily bids

a discretized set of bids $b^1 \ldots b^B$. We define a probability mass function f over this set of bids:

$$f(b^i) = max(g(\frac{b^i + b^{i+1}}{2}) - g(\frac{b^i + b^{i-1}}{2}), \epsilon)Z \qquad (10)$$

for a normalizing constant Z and some small $\epsilon > 0$. The corresponding cumulative distribution function F is now strictly increasing, and $F(b^B) = 1$.

Each time the particle filter needs to draw a new particle p' based on an existing particle p, we generate f and F for each advertiser and then follow the procedure described in Section 3.1. Note that in this case, the advertiser's bid history used to generate the feature vector that is input to the model is based on the bid history stored in the particle p, and not on the (unknown) true bid history.

6. EXPERIMENTS

We now report on experiments that demonstrate the effectiveness of our particle filter for bid estimation. We begin by presenting the experimental setup and describing alternate bid estimation methods against which we compare our particle filter.

6.1 Setup

We evaluate our particle filter in three different settings. For each setting, we use our agent TacTex [7], a top-performing agent from the 2009 TAC/AA competition, as the advertiser we participate as (i.e., the agent whose observations we see and on whose behalf we are estimating bids). The other seven advertiser agents are chosen from the TAC Agent Repository[1], a collection of agent binaries. Different sets of agents are used for each of the three settings. For each setting, we run 50 games. 40 games are used to generate training data, and the remaining 10 are used for testing. For each advertiser, we train a model as described in Section 5.

For testing, we run our particle filter independently for each of the 90 60-day bidding episodes (nine independent episodes per game, as described in Section 4) contained in the test games. In each episode, we initialize each particle by drawing a bid for each advertiser from a histogram of that advertiser's initial bids. Then each day, we update our bid estimates by giving the particle filter the bid rankings, average positions, and TacTex's bid and cpc for that day and performing the update procedure described in Section 3.1.

In our particle filter implementation, we use 2000 particles and did not observe an increase in accuracy from increasing this number. We discretize the bid space into intervals of 0.01, with a maximum bid of 1.1 times the highest bid observed from any advertiser, and we set $\epsilon = 0.0001$.

Our goal in estimating bids is not to track the behavior of a specific advertiser but to get an idea of how much we would need to bid to reach a certain position. We therefore evaluate the performance of our particle filter by comparing our estimate of the nth ranked bid (based on the weighted mean of the particles) with the actual bid for each relevant value of n.

The first setting we consider involves a set of seven different advertiser agents: AstonTAC, QuakTAC, epflagent, MetroClick, Merlion, and two different versions of Schlemazl. Bidding strategies differ considerably between agents. For example, Figure 1 shows the bids of AstonTAC and QuakTAC for one particular episode. Here AstonTAC's bids tend to drift only slightly from day to day, while QuakTAC's bids take larger jumps but show a clear cyclical pattern. Figure 2 shows how the top five bids change each day, illustrating the difficulty of the bid estimation problem. For our second setting we run TacTex against seven copies of QuakTAC, and for our third setting we run TacTex against seven copies of AstonTAC.

6.2 Alternate Bid Estimation Methods

To evaluate the effectiveness of our particle filter, we need to compare its accuracy to other bid estimation methods. The first method we consider is a simple baseline of always estimating the nth ranked bid to be the average nth bid over the training set.

The second method was used in TacTex, so we will call it the TT estimator. Like the particle filter described in this paper, this method is also a form of sequential Bayesian filtering, but there are several important differences. First,

[1] `http://www.sics.se/tac/showagents.php`

instead of using a set of particles to represent a distribution over bids, the TT estimator is a grid-based method, meaning that it explicitly computes a probability mass function over a set of discrete bids. Second, the TT estimator maintains this function independently for each advertiser, instead of estimating a joint distribution over all bids, which requires a number of simplifying assumptions. Third, the TT estimator makes use of a much simpler bidding model than the models learned in Section 5.

The simple bidding model assumes that bids change in one of three ways. First, with probability 0.1, the bid jumps to a random bid. This case covers sudden jumps that are difficult to model. Next, with probability 0.5, the bid changes only slightly from the previous bid. This case reflects the behavior of AstonTAC in Figure 1. The change in bids is modeled under the assumption that the difference in logarithms of successive bids is distributed normally with zero mean. Finally, with probability 0.4, the bid changes according to a similar distribution, but the change is with respect to the bid 5 days ago. This case reflects the behavior of QuakTAC in Figure 1. This model is used for all advertisers.

The TT estimator performs a two step update each day. First, it updates the distribution over each advertiser's bid using the simple bidding model. Second, it multiplies the probability of each bid by the probability that the other advertisers' bids would be consistent with the observed bid ranking given that bid (assuming that the estimated distributions over their bids are correct) and then normalizes. Full details are available in [7].

The third alternate estimator we test is to use our particle filter with the simple bidding model from the TT estimator. We also considered the opposite combination — using the bidding models described in Section 5 with the TT estimator. However, because the TT estimator maintains only bid distributions, and not particles representing bid histories, we do not have the information required to use these bidding models. We tried using the mean of each advertiser's bid on each previous day as the bid history, but results were poor.

6.3 Estimation Results

Table 1 shows the results for all three settings for all bid estimators. For each of the 90 episodes from the 10 test games, we found the root mean squared error of the estimates, and the average RMS error is displayed. We ignored the first five game days in computing these errors so that the errors would not be skewed by start-game effects. (The method of simply using the average bid was especially inaccurate during this period.) In settings 1 and 2, we show the errors of the estimates for the top five bids, since there were nearly always at least five ranked bidders in these settings. In setting 3, however, there were often only three ranked bidders, and so we show three errors.

For all bid estimators, errors were highest on the top ranked bid and generally decreased as the rank increased. This result is expected since the top bid is essentially unbounded above and can fluctuate significantly (as in Figure 2), while lower bids tend to be grouped more tightly. Errors on settings 2 and 3 were much lower than on setting 1. Both QuakTAC and AstonTAC have somewhat predictable bidding patterns and avoid particularly high bids.

Our particle filter with the learned bidder models consistently gave the lowest error of any estimator. In all but one case (setting 3 rank 3) the difference between this error and all other errors was statistically significant ($p < 0.05$) according to a Wilcoxon matched-pairs signed-ranks test. Not surprisingly, using the average bid was worst overall. The performance of the particle filter using the simple bidder model and the TT estimator (which uses the same model) was mixed, with neither clearly outperforming the other. This result is somewhat surprising, since the particle filter is in theory a more principled approach. It may be the case that the deficiencies of the simple bidder model affect each approach differently and that in some cases the TT estimator is more robust.

6.4 Application to Bidding

Finally, while the focus of this paper has been on estimating bids accurately, the goal of this estimation is ultimately to allow an advertiser to set its own bids effectively. We now briefly explore the usefulness of our particle filter when utilized by a full bidding agent. For each setting, we ran 50 games using the original TacTex (which uses the TT estimator to estimate other advertiser's bids and then optimizes with respect to these estimates), then repeated these games using the particle filter with the learned bidder models. Surprisingly, TacTex's score did not improve in setting 1, apparently due to issues with the estimation of other advertisers' spending limits that are beyond the scope of this paper. Fortunately, QuakTAC and AstonTAC do not make significant use of spending limits, so this problem does not impact settings 2 and 3. In setting 2, using the particle filter improved TacTex's score by 452 (from 78,177), and in setting 3, the score improved by 926 (from 82,424). In both cases, the increase was statistically significant ($p < 0.05$) according to a Wilcoxon matched-pairs signed-ranks test. For reference, we ran each set of games again and fed TacTex the true bids of the other advertisers, and the scores in settings 2 and 3 increased by 847 and 1582, respectively, compared to the scores when the TT estimator was used. Thus, the use of our particle filter appears to provide us with a large portion of the gain to be had from improving bid estimation accuracy.

7. CONCLUSION

In this paper we have introduced a particle filter that can be used to estimate the bids of other advertisers in keyword auctions given a periodic ranking of their bids. The key to this particle filter is a method of sampling new particles (representing an updated set of bids) in such a way that the samples are consistent with the observed bid ranking. Additionally, we have described a learning approach to modeling the bidding behavior of other advertisers. In experiments in the TAC/AA domain, the combination of this particle filter and bidder modeling outperforms all other bid estimation methods tested, including the method that was used in the 2009 TAC/AA champion.

There are several areas in which future work is possible. The results show the importance of using accurate bidder models, and there are a number of additional conditional density estimation approaches we could try. Also, we currently only consider the problem of estimating past bids. The next step is to predict future bids, perhaps by using the bidder models to propagate the estimates forward. Testing our bid estimation approach with real world data is another necessary step. Finally, bid estimation is only one of many subproblems faced in designing a successful bidding agent,

Bid Estimator	*Average RMS error per bid estimate for each bid rank*												
	Setting 1					*Setting 2*					*Setting 3*		
	1	2	3	4	5	1	2	3	4	5	1	2	3
average bid	0.678	0.302	0.228	0.176	0.155	0.191	0.135	0.106	0.086	0.079	0.234	0.127	0.178
TT estimator	0.685	0.289	0.190	0.119	0.110	0.203	0.110	0.096	0.085	0.095	0.185	0.198	0.185
PF simple model	0.603	0.304	0.187	0.115	0.089	0.163	0.089	0.080	0.098	0.118	0.206	0.127	0.095
PF learned models	**0.459**	**0.255**	**0.135**	**0.082**	**0.066**	**0.112**	**0.060**	**0.055**	**0.049**	**0.052**	**0.135**	**0.102**	0.092

Table 1: Bid estimate errors for all estimators and settings. Significantly lowest errors in bold.

and fully integrating the methods presented here with other agent components (such as estimating clickthrough rates) remains an important challenge, as does scaling up to handle many simultaneous auctions.

8. ACKNOWLEDGEMENTS

We would like to thank the TAC/AA development team and all who contributed agents to the agent repository. We also thank Doran Chakraborty for assisting with the development of the TacTex agent. This work has taken place in the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. LARG research is supported in part by grants from the National Science Foundation (CNS-0615104 and IIS-0917122), ONR (N00014-09-1-0658), DARPA (FA8650-08-C-7812), and the Federal Highway Administration (DTFH61-07-H-00030).

9. REFERENCES

[1] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, Feb. 2002.

[2] P. Jordan, B. Cassell, L. Callender, and M. Wellman. The Ad Auctions Game for the 2009 Trading Agent Competition. Technical report, 2009.

[3] P. Jordan and M. Wellman. Designing the ad auctions game for the trading agent competition. In *IJCAI 2009 Workshop on Trading Agent Design and Analysis (TADA)*, Pasadena, California, 2009.

[4] C. Kiekintveld, J. Miller, P. R. Jordan, L. F. Callender, and M. P. Wellman. Forecasting market prices in a supply chain game. *Electronic Commerce Research Applications*, 8(2):63–77, 2009.

[5] S. Lahaie, D. Pennock, A. Saberi, and R. Vohra. Sponsored search auctions. In N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*. Cambridge University Press, 2007.

[6] D. Liu, J. Chen, and A. Whinston. Current issues in keyword auctions. In G. Adomavicius and A. Gupta, editors, *Handbooks in Information Systems: Business Computing*. Emerald, 2009.

[7] D. Pardoe, D. Chakraborty, and P. Stone. TacTex09: A champion bidding agent for ad auctions. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, May 2010.

[8] D. Pardoe, D. Chakraborty, and P. Stone. TacTex09: Champion of the first Trading Agent Competition on AdAuctions. Technical Report AI-10-01, Department of Computer Science, The University of Texas, 2010.

[9] R. E. Schapire, P. Stone, D. McAllester, M. L. Littman, and J. A. Csirik. Modeling auction price uncertainty using boosting-based conditional density estimation. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002.

[10] S. Schone. *Auctions In the Electricity Market : Bidding When Production Capacity Is Constrained*. Springer, Berlin, 2009.

[11] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kauffmann, 1999.