

Bidding Heuristics for Ad Auctions with Applications to TAC AA

Jordan Berg, Amy Greenwald, Victor Naroditskiy, and Eric Sodomka

1 Introduction

The Internet advertising market is massive. Google, whose revenue stream is primarily fueled by Internet advertising, boasted revenues near \$24 billion in 2009.¹ Furthermore, in the UK, during the first half of 2009, more money was spent on Internet advertising than on television advertising!²

Most Internet advertising space is sold via ad auctions, which operate as follows: Well in advance, a publisher (e.g., a search engine) solicits bids from advertisers on potential queries. Then, when a user submits one of those queries, the user is shown sponsored search results (i.e., ads) alongside the organic search results. The order in which the ads are displayed is determined on-the-fly by the publisher, who runs an auction using the previously solicited bids. If (and only if) the user clicks on an ad, the corresponding advertiser pays the publisher.

In this environment, both publishers and advertisers face difficult decision problems. The publishers must determine costs-per-click (CPCs), and in which slots to display the winning ads (i.e., how to rank the advertisers, depending on their bids). The advertisers must choose what queries to bid on and what amount to bid, and set budgets.

Work on ad auctions can be divided into two categories: ad auction design, which addresses the publishers' problem (e.g., [1, 2, 6, 7, 14, 19]), and bidding strategy design, which addresses the advertisers' problem. This paper falls into the latter category. Most work on bidding strategies has been carried out in stylized settings where a number of simplifying assumptions are made about the available information (e.g., number of clicks, CPC, and revenue per click are known and do not change over time) and about the decisions the advertisers must make (e.g., there might be a single bidding period or a single query: e.g., [8, 15, 5, 22]). Correspondingly, most bidding strategies have been evaluated based on their theoretical properties.³

In 2009, a simulated market environment for testing bidding strategies in a more realistic setting was released as part of the annual Trading Agent Competition (TAC). The TAC Ad Auctions (AA) game [11] captures many of the difficulties associated with bidding in real ad auctions, such as bidding on multiple queries and setting budgets in the presence of noisy, delayed, and incomplete information. Further, it enables researchers to benchmark various bidding strategies in a com-

petitive environment without the monetary risks associated with experimentation inside a real ad campaign. The work presented in this paper makes use of the TAC AA framework to empirically evaluate autonomous bidding strategies for ad auctions.

We first present bidding strategies for a stylized ad auctions problem, and then for the more complex Trading Agent Competition for Ad Auctions (TAC AA). We decompose the agent's problem into a modeling sub-problem, where we estimate values such as click probability and CPC, and an optimization sub-problem, where we determine what to bid given these estimates. Our optimization algorithms, the focus of this paper, come in two flavors: rule-based algorithms, which can make reasonable decisions even with inaccurate models; and greedy multiple choice knapsack algorithms, which can make better decisions but require more accurate models to do so.

Designing an agent strategy for a complex, dynamic, multi-agent environment is challenging. Perhaps the greatest difficulty stems from the uncertainty in the environment. In the ad auctions domain, other agents' characteristics and strategies, user behavior, and overall market conditions are all uncertain. One standard approach (e.g., [9]) is to treat what is arguably a game-theoretic problem as a decision-theoretic one. That is, an agent can first build models (i.e., predictions about what is to come), and then optimize its decisions given those models.

Taking this approach, (i.e., first predict; then optimize), the success of an optimization routine can hinge on its robustness to inaccurate predictions. A simple optimizer that uses less accurate models may outperform a more sophisticated optimizer that uses more accurate models. We analyze the trade-off between an optimizer's performance and model accuracy in the TAC AA domain. Specifically, we present experiments where the amount of noise in our agents' models is controlled, and we evaluate the performance of various optimizers as a function of noise.

2 Models

Our optimization algorithms depend on four main models: for each day d , for each query q , the predicted (1) number of *impressions* (i.e., number of times an ad is displayed), $i_q \in \mathbb{N}$; (2) probability of a click given an impression, $PrClick_q \in [0, 1]$; (3) probability of a conversion given a click, $PrConv_q \in [0, 1]$; (4) average CPC, $CPC_q \in \mathbb{R}$.

All of these models depend on the agent's decisions—their choices of bids, budgets, and ad types. We can construct models that allow us to reason about outcomes either for placing a bid or for targeting an average position. If targeting an

¹ <http://finance.yahoo.com/q/is?s=G00G&annual>

² <http://news.bbc.co.uk/2/hi/business/8280557.stm>

³ There are, of course, exceptions: e.g., [13].

average position, an additional model is required that maps average positions to bids. In this paper, we use models that reason about bids directly, not average positions.

To make each of these predictions, an agent could either model opponents as individuals or the market as a whole. Because of the limited information the publisher reveals about opponents, we take the latter approach. Our current models are ensembles of regressions built using WEKA,⁴ a pre-existing machine learning toolbox. Because this paper is primarily about optimization, we do not describe our models in detail. For a more comprehensive description of modeling in TAC AA, see Pardoe *et al.* [17].

3 A Stylized Problem

Unlike, for example, second-price auctions [20], deciding how to bid in the TAC AA game,⁵ and in real ad auctions, is difficult. A good starting point, therefore, is to reason about more stylized versions of these problems. We now present such a stylized problem, and then we propose two solutions, one of which is optimal (under certain assumptions) and another of which appears to be a decent approximation, because it performs relatively well in TAC AA.

The simplification we consider is a one-day problem⁶ in which the daily capacity limit is a hard constraint. In other words, an advertiser whose sales reach capacity can no longer participate in the auctions. One possible backstory for this problem is that the advertisers sell products requiring a common perishable component, and they only have a limited amount of that component in stock each day; after this inventory is consumed, no further sales can be made.

The agent's goal in the stylized problem is to choose per-query bids that maximize its profits without exceeding the capacity C . Denoting the profit from bidding $b_q \in \mathbb{R}$ on query $q \in Q$ by $rev_q(b_q) - cost_q(b_q)$ and the corresponding number of sales by $sales_q(b_q)$, the stylized problem is:

$$\begin{aligned} \max_{b \in \mathbb{R}_+^{|Q|}} \sum_{q \in Q} (rev_q(b_q) - cost_q(b_q)) \quad \text{s.t.} \\ \sum_{q \in Q} sales_q(b_q) \leq C \end{aligned}$$

This stylized problem is an instance of the nonlinear knapsack problem (NLK) [10].

Because capacity is the limited resource in this stylized problem, we define *return on investment (ROI)* as follows:

$$ROI_q(b_q) = \frac{rev_q(b_q) - cost_q(b_q)}{sales_q(b_q)} \quad (1)$$

Additionally, *marginal ROI*, the ratio of marginal profit to marginal usage of capacity, is defined as:

$$\text{marginal } ROI_q(b_q) = \frac{rev'_q(b_q) - cost'_q(b_q)}{sales'_q(b_q)} \quad (2)$$

Assuming nonincreasing marginal *ROIs*, and cross-query independence—a decision about how to bid on one query does not impact the *ROI* of another—an optimal solution to our

stylized problem is characterized by the equimarginal principle (EMP) [16], which states that an allocation that maximizes utility (profit) is one for which the return on the last unit of the limited resource (capacity) is equated across all possible uses (queries): i.e., an allocation that maximizes utility is one that equates marginal *ROIs* across uses.

Using the four models presented in Sec. 2, we define the revenue, cost, and sales functions in TAC AA as follows:

$$rev_q(b_q) = sales_q(b_q) USP_q \quad (3)$$

$$cost_q(b_q) = numClicks CPC_q \quad (4)$$

$$sales_q(b_q) = numClicks PrConv_q \quad (5)$$

$$numClicks = i_q PrClick_q \quad (6)$$

In reality, these functions depend on budgets and ad types as well as bids. However, the stylized problem, and our extensions of it, choose only bids, so our agents set budgets and ad types after calculating bids. Further, only our click probability and CPC models actually vary with bids. Our other models are constant across bids.

In TAC AA, these functions are not differentiable, however (see the game specification for details). Hence, we cannot simply take derivatives and then solve for the set of bids that equates marginal *ROIs*. Instead, one approach is to discretize the stylized problem and then attempt to solve the discretized problem optimally by equating marginal *ROIs*. Sec. 3.1 describes this approach. Alternatively, Sec. 3.2 presents a suite of related algorithms that use simple rules to approximate a solution to the continuous version of the stylized problem. The algorithms in the latter set rely less heavily on modeling than the former algorithm.

3.1 Greedy MCKP Algorithm

In this section, we describe an algorithm that roughly equates marginal *ROIs* in a discrete approximation of the stylized problem. Following Zhou and Naroditskiy [21], the discrete approximation can be cast as a 0/1 multiple-choice knapsack problem (MCKP) where submitting the bid b_q in the set B_q corresponds to selecting an item with weight $w_{qb} = sales_q(b_q)$ and value $v_{qb} = rev_q(b_q) - cost_q(b_q)$:

$$\max_{x_{qb}} \sum_{q \in Q} \sum_{b \in B_q} v_{qb} x_{qb} \quad (7)$$

$$\sum_{q \in Q} \sum_{b \in B_q} w_{qb} x_{qb} \leq C \quad (8)$$

$$\sum_{b \in B_q} x_{qb} \leq 1 \quad \forall q \in Q \quad (9)$$

$$x_{qb} \in \{0, 1\} \quad \forall q \in Q \text{ and } \forall b \in B_q \quad (10)$$

Eq. (7) is the objective function, which is to maximize the sum of the profits across all queries. Eq. (8) is the capacity constraint. Eq. (9) is the multiple choice constraint, which states that only one bid can be taken from each bid set. Eq. (10) is the 0/1 constraint.

MCKP is typically solved greedily [12]. GreedyMCKP converts an instance of MCKP into a standard knapsack problem (KP) by creating *incremental* items, and then solves the resulting KP greedily. Incremental items are taken in non-increasing order of *efficiency*—defined as value divided by

⁴ <http://www.cs.waikato.ac.nz/ml/weka/>

⁵ A detailed description of the TAC AA game scenario can be found here: <http://tac.eecs.umich.edu/about-the-game/>.

⁶ Alternatively, we could assume independence across days.

weight—until the knapsack has reached its capacity or there are no items left with positive efficiencies. The incremental items are so-called because taking the first k of them in the resulting KP is equivalent to taking the k th item in the original MCKP: i.e., a solution to this KP can be immediately interpreted as a solution to the original MCKP.

Given an instance of MCKP, with an item set $\{(w_{qb}, v_{qb}) \mid \forall b \in B_q\}$ for each query q , we construct a set of incremental items in three steps. In the first step, we sort the items in nondecreasing order by weight, redefining (w_{qi}, v_{qi}) to be the weight and value of the i th lightest item for query q . In the second step, we remove *dominated* and *LP-dominated* items [21]. From the remaining items, in the third step, we create incremental items $(\bar{w}_{qi}, \bar{v}_{qi})$, where $\bar{w}_{q1} = w_{q1}$, $\bar{v}_{q1} = v_{q1}$, $\bar{w}_{qj} = w_{qj} - w_{q,j-1}$, and $\bar{v}_{qj} = v_{qj} - v_{q,j-1}$, for $2 \leq j \leq n$.

To see that this greedy algorithm roughly equates marginal *ROIs* in the discrete approximation, let $(\bar{w}_{qi^*}, \bar{v}_{qi^*})$ denote the last incremental item taken in a greedy solution. By the greedy nature of this solution, all incremental items with efficiencies above $\frac{\bar{v}_{qi^*}}{\bar{w}_{qi^*}}$ are also taken. In other words, the efficiency of the last incremental item taken in each query is roughly equal to $\frac{\bar{v}_{qi^*}}{\bar{w}_{qi^*}}$. But, the efficiency of an incremental item is the discrete counterpart of marginal *ROI*.

3.2 Rule-Based Algorithms

Applying GreedyMCKP to the stylized bidding problem requires estimating revenue, cost, and sales functions, which in TAC AA depend on the four models discussed in Sec. 2. These models are built in a dynamic, uncertain, multi-agent environment. Not surprisingly, decisions that are made using grossly inaccurate models can be very poor.

Next, we present a set of rule-based bidding heuristics that are less reliant on our models. Specifically, the rule-based bidders only rely on the conversion probability model.

Because revenue, sales, and costs are not differentiable in TAC AA, we cannot directly search for the bids at which marginal *ROIs* are equated across queries. Instead, our rule-based bidders search for the bids at which some proxy of marginal *ROI*, like *ROI* itself, is equated across queries.

More specifically, instead of solving a system of equations like this one (for three queries, q , q' , and q''):

$$\begin{aligned} \text{marginal } ROI_q(b_q) &= \text{marginal } ROI_{q'}(b_{q'}) \\ \text{marginal } ROI_q(b_q) &= \text{marginal } ROI_{q''}(b_{q''}) \\ \text{sales}_q(b_q) + \text{sales}_{q'}(b_{q'}) + \text{sales}_{q''}(b_{q''}) &= C \end{aligned}$$

the first rule-based bidder we present, *EquateROI*, iteratively approximates a solution to a system of equations like this one (again, for only three queries):

$$\begin{aligned} ROI_q(b_q) &= ROI_{q'}(b_{q'}) \\ ROI_q(b_q) &= ROI_{q''}(b_{q''}) \\ \text{sales}_q(b_q) + \text{sales}_{q'}(b_{q'}) + \text{sales}_{q''}(b_{q''}) &= C \end{aligned}$$

Intuitively, this latter set of equations states that *ROIs* across queries are all equal to some *targetROI*, and that sales summed over all queries equals capacity.

Solving this system exactly requires estimating revenue, cost, and sales functions—precisely what we set out not to

do! Instead, our *EquateROI* bidder approximates this computation incrementally: it increases the value of *targetROI* if yesterday's sales exceeded capacity and decreases it otherwise.⁷ Bids are then selected by first calculating the CPC corresponding to the current *targetROI*, and then converting this CPC into a bid. Details follow.

Recall the definition of *ROI* and the definitions of revenue, cost, and sales from Eqs. (3), (4), and (5):

$$ROI_q(b_q) = \frac{\text{rev}_q(b_q) - \text{cost}_q(b_q)}{\text{sales}_q(b_q)} = USP_q - \frac{CPC_q}{PrConv_q}$$

Solving for CPC yields:

$$CPC_q = (USP_q - ROI_q(b_q))PrConv_q$$

Therefore, for a desired *targetROI*, we have $CPC_q = (USP_q - \text{targetROI})PrConv_q$.

At this point, we could invoke the *cpc2bid*⁸ model with CPC_q to obtain the bid b_q for which $ROI_q(b_q) = \text{targetROI}$. Instead, we limit the rule-based algorithms reliance on models even further; they simply bid $CPC_q + \epsilon$.

Equating *ROIs* is a natural rule but not the only one that comes to mind. For instance, one could also consider equating profit margins (*PM*): i.e., profit per order dollar. It is also common to look at cost per order dollar (*CPOD*) as a measure of the success of a marketing campaign. Another possibility is profit per cost dollar (*PPCD*). It turns out that the strategies that result from trying to equate any of these three metrics are equivalent [3].

Among the three corresponding and equivalent strategies, we evaluate the performance of only *EquatePM*. *EquatePM* takes the same steps to equate *PMs* that *EquateROI* takes to equate *ROIs*. In particular, it adjusts a target *PM* based on yesterday's sales and capacity, and then converts that target *PM* into bids as follows:

$$PM_q(b_q) = \frac{\text{rev}_q(b_q) - \text{cost}_q(b_q)}{\text{rev}_q(b_q)} \quad (11)$$

$$= 1 - \frac{CPC_q(b_q) \text{numClicks}_q(b_q)}{\text{numClicks}_q(b_q) PrConv_q USP_q} \quad (12)$$

$$= 1 - \frac{CPC_q(b_q)}{PrConv_q USP_q} \quad (13)$$

$$CPC_q = (1 - \text{targetPM}) PrConv_q USP_q \quad (14)$$

In Sec. 5, we evaluate the performance of GreedyMCKP and the rule-based bidders in TAC AA, a more realistic setting than the stylized problem. But first we extend the GreedyMCKP algorithm to solve a more complex problem.

4 The TAC AA Bidding Problem

In this section, we relax two key assumptions present in the stylized problem—the existence of a hard capacity constraint and the one-shot nature of the decisions. These relaxations lead to an extended problem that more closely models the TAC AA bidding problem and bidding in real-world ad auctions than the stylized problem, because it models cross-query and cross-day dependencies.

⁷ This simple strategy of adjusting *ROI* in the direction of some target *ROI* has been analyzed previously in the context of mechanism design for ad auctions [5].

⁸ The *cpc2bid* model is simply the inverse of the $CPC_q(b_q)$ model described in Sec. 2.

In the stylized problem, there are no cross-query dependencies: an agent's *ROI* (revenue, cost, and sales) on query q depends only on its bid on query q , not on its decisions about other queries. Also, there are no cross-day dependencies: an agent's *ROI* on day d depends only on its bids on day d , not on its future decisions. But in the TAC AA bidding problem, there are both cross-query and cross-day dependencies: an agent's *ROI* on query q depends on its decisions about other queries on other days.

The cross-query and cross-day dependencies in TAC AA arise from a penalty function that links decisions across queries and days. There is no hard limit on an agent's sales potential. On the contrary, there is a soft limit: i.e., there is a point (so-called "capacity") beyond which a penalty is paid for additional sales. More specifically, once total sales over the last W days rises above capacity, conversions in all queries become less likely. This choice of penalty structure can be motivated as follows: when an agent sells too many products in a short period of time, it may hit a production or shipping bottleneck. Users who then click on an ad may be notified that the product of interest is on backorder, which may make them less likely to convert.

This section is organized as follows. First, we study a problem with only cross-query dependencies; in particular, total sales for only the current day are factored into the penalty function (i.e. $W = 1$). We argue that this problem is an instance of what we call the *penalized* MCKP (PMCKP) [4], and we design greedy heuristics (building on GreedyMCKP) to "solve" PMCKP. We then enhance that problem with cross-day dependencies by requiring that sales from the last $W > 1$ days be factored into the penalty function, so that sales on a given day affect the penalties on future days. We use a simple heuristic, built on top of our "solutions" to PMCKP, to "solve" this latter, more complex, problem.

4.1 Cross-Query Dependencies

In our first modification of the stylized problem, the capacity C is not a hard limit, but rather a point beyond which additional sales are penalized. In particular, when an agent has converted i units beyond its capacity, the probability of a further conversion given a click is scaled down by λ^i , for $0 < \lambda < 1$. But the order in which various users input search queries and possibly convert is governed by a stochastic process, so the precise sequence of conversion probabilities that are realized in each query as users search, click, and convert is uncertain. Consequently, we calculate *expected* conversion probability, $\overline{PrConv}_q(\rho, \kappa)$, as follows:

$$\overline{PrConv}_q(\rho, \kappa) = \frac{\pi_q}{\kappa} \sum_{i=\rho}^{\kappa} \lambda^{\max\{0, i-C\}} \quad (15)$$

Here, ρ is the initial capacity used, κ is the final capacity used, and π_q is a query-specific "baseline" conversion probability (see the game specification for details).⁹

Recall that the stylized problem could be cast as an instance of MCKP with $w_{qb} = \text{sales}_q(b_q)$ and $v_{qb} = \text{rev}_q(b_q) -$

$\text{cost}_q(b_q)$. Following this same logic, weights and values in this modified problem would be defined similarly, based on *expected* conversion probability: i.e.,

$$v_{qb}(\rho, \kappa) = \overline{\text{rev}}_q(b_q, \rho, \kappa) - \text{cost}_q(b_q) \quad (16)$$

$$w_{qb}(\rho, \kappa) = \overline{\text{sales}}_q(b_q, \rho, \kappa) \quad (17)$$

$$\overline{\text{rev}}_q(b_q, \rho, \kappa) = \overline{\text{sales}}_q(b_q, \rho, \kappa) \overline{USP}_q \quad (18)$$

$$\overline{\text{sales}}_q(b_q, \rho, \kappa) = i_q \overline{PrClick}_q \overline{PrConv}_q(\rho, \kappa) \quad (19)$$

(The cost function is unchanged.)

Using these weights and values, and dropping the hard capacity constraint, we arrive at the following *penalized* version of MCKP (PMCKP) [4]:

$$\max_x \sum_{q \in Q} \sum_{b \in B_q} v_{qb}(\rho, \kappa) x_{qb} \quad (20)$$

$$\sum_{b \in B_q} x_{qb} \leq 1 \quad \forall q \in Q \quad (21)$$

$$x_{qb} \in \{0, 1\} \quad \forall q \in Q \text{ and } \forall b \in B_q \quad (22)$$

In this, the one-shot version of PMCKP, $\rho = 0$. The only remaining question, then, is how to calculate κ . By definition, κ is the sum of the weights across queries and bids. But on the other hand, *weights depend on κ* : i.e., $\kappa = \sum_{qb} w_{qb}(\rho, \kappa) x_{qb}$. In our MCKP-based bidders, we solve for κ iteratively. It appears that this iterative approach always converges, but this claim warrants further theoretical investigation.

Next, we describe three algorithms designed to solve PMCKP. The first is a near-optimal but slow algorithm. The other two trade off optimality for speed to varying degrees.

Exhaustive Greedy PMCKP For any given ρ (0, for now) and κ , values and weights are known. Hence, one could solve PMCKP by undertaking an exhaustive search for the best possible value of κ , as follows: first, discretize the space of possible κ values; then for each possible value of κ , set values and weights accordingly and solve the ensuing MCKP using GreedyMCKP; finally, after trying all possible κ values, return the bids returned by the GreedyMCKP algorithm whose κ level resulted in the highest profit. This algorithm is slow, but exactly how slow depends on the discretization factor.

Dynamic Greedy PMCKP A more computationally efficient approach is to choose κ dynamically. Recalling GreedyMCKP, observe that for PMCKP we can not simply eliminate all dominated items in a preprocessing step, because values and weights now vary with κ , so items that are dominated when κ is zero might be undominated when κ is positive, and vice versa. Consequently, we must modify GreedyMCKP to eliminate dominated items each time another item is taken, because taking items causes the capacity used to change.

DynamicPMCKP, a dynamic version of GreedyMCKP, finds the next undominated item and creates the next incremental item for each query. It then searches across all queries to find an incremental item of maximal efficiency, and so long as that efficiency is positive, that incremental item is taken. This process repeats until an incremental item is encountered whose efficiency is nonpositive or no incremental items remain, at which point the incremental items that were taken are converted back to (regular) items.

⁹ More precisely, we feed this expected conversion probability into another (nonlinear) function η which massages this value based on the agent's manufacturer specialty bonus (MSB): i.e., $\overline{PrConv}'_q(\rho, \kappa) = \eta(\overline{PrConv}_q(\rho, \kappa), \text{MSB})$.

Hybrid Greedy PMCKP Eliminating dominated items requires sorting, meaning there is an $O(n \log n)$ operation inside the main loop of DynamicPMCKP. HybridPMCKP ignores the fact that the set of undominated items can change from one iteration to the next. It eliminates dominated items once, as a preprocessing step, before entering its main loop. This order then remains constant throughout execution, but values and weights are updated in the main loop as the capacity used changes, which yields corresponding changes in incremental efficiencies. DynamicPMCKP is $O(n^2 \log n)$, where n is the total number of items across item sets, while HybridPMCKP is only $O(n^2)$ [4].

4.2 Cross-Day Dependencies

The multi-day TAC AA bidding problem is presented formally in Berg, *et al.* [3].

In the multi-day problem, it may not be optimal to maximize total value on the current day, because placing the corresponding bids could adversely impact future value. One could develop a multi-day optimization algorithm (based on dynamic programming, for example) to solve this problem optimally. Instead, we extend our PMCKP algorithms with a simple heuristic that is computationally inexpensive and still appears to make profitable decisions.

First, we assume that our models, built only for the current day, are applicable as they stand every day into the future. Second, the only bid vectors that we consider are constant across days; that is, when we consider placing a bid, we consider placing that bid today and every day into the future. Under these assumptions, we calculate the total value associated with each potential bid, and then choose one using one of our greedy PMCKP algorithms.

5 Evaluation

We now evaluate the bidding heuristics described above. We first present our controlled testing framework, then describe our experimental design, and finally present our results.

Testing Framework We refer to our controlled testing framework as the *optimization challenge*. Two key features of the optimization challenge are: (1) the game-theoretic complexities of the TAC AA game are not present, and (2) the degree of accuracy of an agent’s models can be controlled.

The optimization challenge takes as input a previously played TAC AA game,¹⁰ and replays that game with one of the original agents, say A , replaced by an agent, say B , the latter of which employs the bidding algorithm we are evaluating. Each day, B receives the information that A received. Based on this information, B makes its decisions, and then that day is re-simulated with these new decisions plus the old decisions made by the other agents still in the game ($\neg A$). Note that since it reads opponent behavior from a past game log, the optimization challenge removes the ability for relevant opponent responses, given the new agents’ decisions.

Each bidding algorithm requires models to make its decisions. These models can be the agent’s actual models that make predictions based on the daily information received. Or, they can be artificially improved models created by simulating

the agent placing various bids, since the opponents’ decisions are known. Or, they can be deliberately noisy models.

Experimental Design We run tests with the bidding algorithms designed to solve the stylized problem (GreedyMCKP, EquateROI, and EquatePM), as well as those designed to solve the more realistic problem (ExhaustivePMCKP, DynamicMCKP, and HybridPMCKP). Using the TAC AA game server in conjunction with the optimization challenge described above, we perform the following tests:

1. First, we evaluate performance in actual TAC AA games. A subset of agents play 60 games against the most competitive set of agents available in the TAC agent repository.¹¹
2. Second, we evaluate the performance of our bidding heuristics varying the degrees of model accuracy. In this paper, we focus on the $PrClick_q$ and CPC_q models. Using the optimization challenge, we run HybridPMCKP and EquateROI on the 2009 TAC AA finals games, evaluating how performance changes as a single model becomes less accurate.

Results Our results from running 60 TAC AA games are shown in Table 1. From these games, we observe that HybridPMCKP¹² and EquateROI are both finishing in 2nd place against the highest-performing agents in the TAC agent repository. Also of note is that the profit ordering of opponents is generally consistent across each set of games; hence, the decisions of a single agent do not appear to have a strong impact on the performance of others. This provides some initial evidence that the optimization challenge, which disallows opponent responses, is representative of a TAC AA game.

Agent	Set 1	Set 2	Set 3
TacTex	80.76	79.86	81.84
HybridPMCKP	77.83		
EquateROI		75.67	
astonTAC	76.30	75.32	77.25
munsey	73.41	72.40	72.01
epflagent	72.43	73.05	72.46
EquatePM			69.63
QuakTAC	70.61	70.53	68.38
MetroClick	70.15	68.73	69.10
mertacor	68.31	68.08	67.63

Table 1. Average Performance (60 actual games with best repository agents)

We use the optimization challenge to evaluate the performance of algorithms with different levels of model accuracy (Figure 1). We run HybridPMCKP and EquateROI agents using their standard models, adding varying amounts of Gaussian noise to one model at a time. Because the rule-based algorithms do not use the $PrClick$ or CPC models, EquateROI’s profits are constant regardless of the level of model accuracy.

In these experiments HybridPMCKP performed better than EquateROI with our actual models but became worse as we added more noise. Also, HybridPMCKP is more robust to noise

¹¹ Repository agents are located at <http://www.sics.se/tac/showagents.php>.

¹² For experimental evidence that HybridPMCKP performs as well as DynamicMCKP, which in turn performs as well as ExhaustivePMCKP, see Berg *et al.* [3].

¹⁰ Logs of competition games are publicly available at <http://aa.tradingagents.org/servers/>

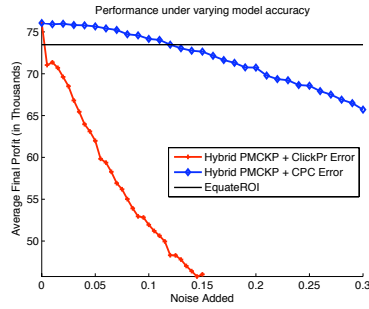


Figure 1. Agent profits with noise added to the click probability and CPC models.

in its *CPC* model than its *PrClick* model. This suggests that improving our *PrClick* model would be more valuable than improving our *CPC* model. Using this framework, an advertiser can evaluate how well any number of optimization algorithms would perform given different levels of model accuracy, and can identify the threshold point at which it would be beneficial to switch to a more model-heavy algorithm.

6 Conclusion

In this paper, we reviewed two known bidding algorithms [5, 21] that were proposed as solutions to a stylized ad auctions problem, and then we extended these algorithms to the Trading Agent Competition for Ad Auctions (TAC AA). We also designed an optimization challenge: i.e., a controlled testing framework that isolates the optimization component of an agent's strategy (as opposed to the modeling component, which is isolated in a prediction challenge [18]). We showed that performance in this optimization challenge is representative of performance in actual TAC AA games. We then used the optimization challenge to evaluate our bidding algorithms' performances with respect to varying degrees of model accuracy. We found that there is no significant difference in performance between an exhaustive algorithm and our greedy algorithms, and we also found that a simple rule-based algorithm can successfully compete in this domain against our greedy algorithms, but its relative performance will likely decrease as model accuracy improves.

In future work, we plan to develop algorithms which can handle additional features of real-world ad auctions that are not modeled in TAC AA, such as reputation or branding effects, which may cause an increased bid in one query to have a positive effect on the value of another query.

ACKNOWLEDGMENTS

A similar version of this paper [3] was presented at TADA 2010. Many students contributed their time and energy to this research, including Max Barrows, Hang Chen, Carleton Coffrin, Donnie Kendall, Sam Pucci, and Cathy Zhang. Thanks also to Bradd Libby, of The Search Agency, for many valuable brainstorming sessions. This work was supported by NSF Grant #CCF-0905234.

REFERENCES

- [1] G. Aggarwal, A. Goel, and R. Motwani. Truthful auctions for pricing search keywords. In *Proc. ACM EC*, pages 1–7, June 2006.
- [2] S. Athey and G. Ellison. Position auctions with consumer search. *Working Paper*, 2007.
- [3] J. Berg, A. Greenwald, V. Naroditskiy, and E. Sodomka. A first approach to autonomous bidding in ad auctions. In *EC 2010 Workshop on Trading Agent Design and Analysis*, June 2010.
- [4] J. Berg, A. Greenwald, V. Naroditskiy, and E. Sodomka. A knapsack-based approach to bidding in ad auctions. In *In the 19th European Conference on Artificial Intelligence (ECAI)*, 2010.
- [5] C. Borgs, J. Chayes, O. Etesami, N. Immorlica, K. Jain, and M. Mahdian. Dynamics of bid optimization in online advertisement auctions. In *Proc. WWW*, pages 531–540, 2007.
- [6] C. Borgs, J. T. Chayes, N. Immorlica, M. Mahdian, and A. Saberi. Multi-unit auctions with budget-constrained bidders. In *Proc. ACM EC*, pages 44–51, 2005.
- [7] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. NBER Working Paper No. 11765, 2005.
- [8] J. Feldman, S. Muthukrishnan, M. Pal, and C. Stein. Budget optimization in search-based advertising auctions. In *ACM EC*, pages 40–49, 2007.
- [9] A. Greenwald, V. Naroditskiy, and S. J. Lee. Roxybot-06: Stochastic prediction and optimization in TAC travel. *Artificial Intelligence Research*, 36:513–546, 2009.
- [10] D. S. Hochbaum. A nonlinear knapsack problem. In *Operations Research Letters*, volume 17, pages p. 103–110(8), April 1995.
- [11] P. Jordan and M. Wellman. Designing an ad auctions game for the trading agent competition. In *Workshop on Trading Agent Design and Analysis*, July 2009.
- [12] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, Berlin, Germany, 2004.
- [13] B. Kitts and B. J. LeBlanc. Optimal bidding on keyword auctions. *Electronic Markets*, 14(3):186–201, 2004.
- [14] A. Mehta, A. Saberi, U. V. Vazirani, and V. V. Vazirani. Adwords and generalized on-line matching. In *Proc. FOCS*, pages 264–273, 2005.
- [15] S. Muthukrishnan, M. Pál, and Z. Svitkina. Stochastic models for budget optimization in search-based advertising. In *WINE*, pages 131–142, 2007.
- [16] T. Odean, V. Naroditskiy, A. Greenwald, and J. Donaldson. Marginal bidding: An application of the equimarginal principle to bidding in tac scm. In *Workshop on Trading Agent Design and Analysis*, July 2007.
- [17] 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 Multi-agent Systems (AAMAS 2010)*, 2010.
- [18] D. Pardoe and P. Stone. The 2007 tac scm prediction challenge. In *AAAI 2008 Workshop on Trading Agent Design and Analysis*, 2008.
- [19] H. R. Varian. Position auctions. *International Journal of Industrial Organization*, 25(6):1163–1178, December 2007.
- [20] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.
- [21] Y. Zhou and V. Naroditskiy. Algorithm for stochastic multiple-choice knapsack problem and application to keywords bidding. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 1175–1176, New York, NY, USA, 2008. ACM.
- [22] Y. Zhou and V. Naroditskiy. Algorithm for stochastic multiple-choice knapsack problem and keywords bidding. In *WWW08: Workshop on Targeting and Ranking for Online Advertising*, 2008.