A First Approach to Autonomous Bidding in Ad Auctions

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

ABSTRACT

We present autonomous bidding strategies for ad auctions, first for a stylized problem, and then for the more realistic Trading Agent Competition for Ad Auctions (TAC AA)—a simulated market environment that tries to capture some of the complex dynamics of bidding in ad auctions.

We decompose the agent's problem into a modeling sub-problem, where we estimate values such as click probability and cost per click, 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.

We evaluate agent performance in the game-theoretic TAC AA domain, as well as in a more controlled testing framework that decouples the modeling and optimization sub-problems. Although decision-theoretic in nature, we argue that it is nonetheless reasonable to use our controlled testing framework to determine which optimization algorithm is most suitable for an agent with a given amount of model error, and which model improvements would lead to the greatest gain in overall agent performance.

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 or a newspaper's web site) 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 organic 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. Finally, if the user clicks on an ad, the corresponding advertiser pays the publisher some amount.

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., [2, 21, 1, 5, 6, 16]), 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., [7, 17, 4, 24]). 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 competitive 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.

More specifically, we present autonomous bidding strategies for ad auctions, first for a stylized 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 cost per click, 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

[1] `http://finance.yahoo.com/q/is?s=GOOG&annual`

[2] `http://news.bbc.co.uk/2/hi/business/8280557.stm`

[3] There are, of course, exceptions: e.g., [14].

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, multiagent environment is challenging. Perhaps the greatest difficulty stems from the uncertainty in the environment. In the ad auctions domain, there is uncertainty about many things, including other agents' characteristics and strategies, user behavior, and overall market conditions. One standard approach (e.g., [9]) is to treat what is arguably a game-theoretic problem decision-theoretically. 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 tradeoff 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.

Following the work of Pardoe [20], we implemented a TAC AA *prediction challenge*. We use this challenge to determine model accuracy. Given model accuracy, the experimental methodology we present here, which we call an *optimization challenge*, can be used to identify successful optimizers. Interestingly, it can also be used to identify the set of models whose improvement would lead to the greatest gain in overall agent performance. The optimization challenge for TAC AA extends the experimental methodology developed in Greenwald *et al.* [8] for the TAC Travel domain. Hence, we contend that our approach has general applicability.

The remainder of this paper proceeds as follows: In Sec. 2, we describe the TAC AA game in more detail. In Sec. 3, we describe the models that are input to our optimization algorithms, which estimate values such as click probability and cost per click. In Sec. 4, we present a stylized version of the ad auctions problem, and describe optimization algorithms to solve this problem: a model-heavy greedy multiple choice knapsack algorithm (Sec. 4.1) and various model-light rule-based algorithms (Sec. 4.2). In Sec. 5, we present extensions to the stylized ad auctions problem that makes the problem more realistic, and describe extensions to our stylized algorithms that solve these more complex problems. In Sec. 6, we describe our controlled testing framework and use it to evaluate the performance of our optimization algorithms in both the TAC AA game and in more controlled experiments. We provide concluding remarks in Sec. 7.

2. AD AUCTIONS GAME

We begin with a brief summary of the relevant aspects of the TAC AA game scenario.[4] There are eight agents, each of which represents an online advertiser and seeks to maximize profits over the course of $D = 60$ days.

The ad ranking mechanism for TAC AA varies across games along a continuum of *rank-by-bid* and *rank-by-revenue* mechanisms [15]. (The particular choice of mechanism is revealed to the agents at the start of each game.)

Each advertiser in TAC AA is an online electronics retailer that sells nine different products, each one specified by a manufacturer (Flat, Lioneer, and PG) and a component (TV, Audio, and DVD). Each advertiser is assigned a manufacturer and a component specialty, which raise the unit sales price (USP) and the conversion probability, respectively, of users purchasing the corresponding products and clicking on the corresponding ads.

Correspondingly, users query search terms that include: both a manufacturer and a component, either a manufacturer or a component, or neither a manufacturer nor a component. There are 16 possible search queries in all. After a user submits a search query, the user's behavior follows a variant of the *cascade model* [13]. Furthermore, users are distributed among *non-searching*, *searching*, and (already) *transacted* states, and transition among states according to probabilities specified by a Markov chain. This rich model of user behavior is arguably one of the aspects of TAC AA that makes it more realistic than competing models.

Each day, the agents send the following information to the publisher: For each query, a *bid* representing the maximum price the advertiser is willing to pay per click for that query; an *ad type*—targeted (e.g., "We sell Lioneer TVs.") or generic (e.g., "We sell electronics."); a per-query *budget*, which specifies the total amount the advertiser is willing to spend on a particular query on a single day; and an *overall budget*, which bounds the total amount the advertiser is willing to spend across all queries on a single day.

In addition to their manufacturer and component specialties, each advertiser is also assigned a *capacity* limit, which is an upper bound on the number of sales over the last $W = 5$ days that can be made without penalty. If sales exceed this capacity, then the conversion probabilities of subsequent users who click on an ad decrease. Advertisers must account for this penalty when making bidding decisions or they will end up paying for clicks that have little chance of generating any revenue. Capacity is the main aspect of the TAC AA game that links agent decisions across queries and days.

3. 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 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.

Fortunately, the publisher sends the agents daily reports summarizing past activity. Agents do not receive detailed information about their opponents' bids and budgets, (though they can try to infer that information; see Pardoe [19]). They are told only the following: for each query, all agents' average positions, all agents' ad types, their own average CPC, and the number of impressions, clicks, and conversions they received. These reports are two days old, and they pertain only to the single point at which the agents'

[4]The complete game specification is available at `http://aa.tradingagents.org/`.

bid was placed.

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,[5] 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.* [19].

4. A STYLIZED PROBLEM

Unlike, for example, second-price auctions [22], deciding how to bid in the TAC AA game, and in real ad auctions, is difficult. A good starting point, therefore, is to 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 problem[6] 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:

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

$$\sum_{q \in Q} sales_q(b_q) \leq C$$

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:

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

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

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

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) [18], 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.

[5] `http://www.cs.waikato.ac.nz/ml/weka/`

[6] Alternatively, we could assume independence across days.

Using the four models presented in Sec. 3, we define the revenue, cost, and sales functions in TAC AA as follows:

$$rev_q(b_q) = sales_q(b_q)\, USP_q \tag{3}$$

$$cost_q(b_q) = numClicks\, CPC_q \tag{4}$$

$$sales_q(b_q) = numClicks\, PrConv_q \tag{5}$$

$$numClicks = i_q\, PrClick_q \tag{6}$$

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.

These functions are not differentiable, however (see the TAC AA specification[7] 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. 4.1 describes this approach. Alternatively, Sec. 4.2 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.

4.1 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 [23], 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 $v_{qb} = rev_q(b_q) - cost_q(b_q)$:

$$\max_{x_{qb}} \sum_{q \in Q} \sum_{b \in B_q} v_{qb} x_{qb} \tag{7}$$

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

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

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

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 nonincreasing order of *efficiency*—defined as value divided by 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 kth 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

[7] `http://aa.tradingagents.org`

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 ith lightest item for query q. In the second step, we remove *dominated and LP-dominated* items. 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 $2 \leq j \leq n$.

An item a *dominates* another item b if $w_a \leq w_b$ and $v_a > v_b$. For example, item (1,100) dominates item (5,20); there is never a reason to take the second item when you can take the first. Items a and b *LP-dominate* item c if $w_a < w_c < w_b$, $v_a < v_c < v_b$, and $\frac{v_b - v_c}{w_b - w_c} \geq \frac{v_c - v_a}{w_c - w_a}$. For example, given items (10,25), (50,100), and (40,60), the first and second LP-dominate the third because the linear combination $.5(10, 25) + .5(50, 100) = (30, 62.5)$ dominates (40,60).

To see that this greedy algorithm roughly equates marginal *ROI*s 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.

4.2 Rule-Based Algorithms

Applying GreedyMCKP to the stylized bidding problem requires estimating revenue, cost, and sales functions, which for TAC AA depends on the four models discussed in Sec. 3. But these predictions are made in a dynamic, uncertain, multiagent 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 *ROI*s 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''):

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

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):

$$\begin{aligned}
&ROI_q(b_q) = ROI_{q'}(b_{q'}) \\
&ROI_q(b_q) = ROI_{q''}(b_{q''}) \\
&sales_q(b_q) + sales_{q'}(b_{q'}) + sales_{q''}(b_{q''}) = C
\end{aligned}$$

Intuitively, this latter set of equations states that *ROI*s 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.[8] 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):

$$ROI_q(b_q) = \frac{rev_q(b_q) - cost_q(b_q)}{sales_q(b_q)} = USP_q - \frac{CPC_q}{PrConv_q}$$

Solving for CPC yields:

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

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

At this point, we could invoke the *cpc2bid*[9] model with CPC_q to obtain the bid b_q for which $ROI_q(b_q) = targetROI$. Instead, we limit the rule-based algorithms reliance on models even further; they simply bid $CPC_q + \epsilon$.

Pseudocode for the EquateROI bidder is shown in Alg. 1, where bids for day $d+1$ are set based on the *targetROI* on day d and the observed sales on that day.

Algorithm 1 EquateROI

Input: $sales(d)$, $targetROI(d)$, C
Output: $\forall q, bid_q$

```
if sales(d) > C then
  targetROI(d+1) = targetROI(d) * INC_ROI
else if sales(d) < C then
  targetROI(d+1) = targetROI(d) / INC_ROI
end if
for all q do
  CPC_q = (USP_q - targetROI(d+1)) * PrConv_q
  bid_q = CPC_q + ε
end for
```

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.

To demonstrate this equivalence, we express each metric in terms of *PM*:

$$PM_q(b_q) = \frac{rev_q(b_q) - cost_q(b_q)}{rev_q(b_q)} = 1 - \frac{cost_q(b_q)}{rev_q(b_q)} \quad (11)$$

$$CPOD_q(b_q) = \frac{cost_q(b_q)}{rev_q(b_q)} = 1 - PM_q(b_q) \quad (12)$$

$$PPCD_q(b_q) = \frac{rev_q(b_q) - cost_q(b_q)}{cost_q(b_q)} = \frac{PM_q(b_q)}{1 - PM_q(b_q)} \quad (13)$$

Now suppose *PM*s have been equated across queries at the level *targetPM*. By Eq. (12), the *CPOD* in each query is the same as well: $CPOD_q(b_q) = 1 - PM_q(b_q) = 1 - targetPM$. By the same argument, when any of these targets is equated across queries, the other ones are equated as well.

[8] 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 [4].

[9] The *cpc2bid* model is simply the inverse of the $CPC_q(b_q)$ model described in Sec. 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:

$$PM_q(b_q) = \frac{rev_q(b_q) - cost_q(b_q)}{rev_q(b_q)} \quad (14)$$

$$= 1 - \frac{CPC_q(b_q) numClicks_q(b_q)}{numClicks_q(b_q) PrConv_q USP_q} \quad (15)$$

$$= 1 - \frac{CPC_q(b_q)}{PrConv_q USP_q} \quad (16)$$

$$CPC_q = (1 - targetPM) PrConv_q USP_q \quad (17)$$

In Sec. 6, 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.

5. 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.

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) [3], 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.

5.1 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:

$$\overline{PrConv_q}(\rho, \kappa) = \frac{\pi_q}{\kappa} \sum_{i=\rho}^{\kappa} \lambda^{\max\{0, i-C\}} \quad (18)$$

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).[10]

Recall that the stylized problem could be cast as an instance of MCKP with $w_{qb} = sales_q(b_q)$ and $v_{qb} = rev_q(b_q) - cost_q(b_q)$. Following this same logic, weights and values in this modified problem would be defined similarly, based on *expected* conversion probability: i.e.,

$$v_{qb}(\rho, \kappa) = \overline{rev_q}(b_q, \rho, \kappa) - cost_q(b_q) \quad (19)$$

$$w_{qb}(\rho, \kappa) = \overline{sales_q}(b_q, \rho, \kappa) \quad (20)$$

where

$$\overline{rev_q}(b_q, \rho, \kappa) = \overline{sales_q}(b_q, \rho, \kappa) USP_q \quad (21)$$

$$\overline{sales_q}(b_q, \rho, \kappa) = i_q PrClick_q \overline{PrConv_q}(\rho, \kappa) \quad (22)$$

(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) [3]:

$$\max_x \sum_{q \in Q} \sum_{b \in B_q} v_{qb}(\rho, \kappa) x_{qb} \quad (23)$$

$$\sum_{b \in B_q} x_{qb} \leq 1 \quad \forall q \in Q \quad (24)$$

$$x_{qb} \in \{0, 1\} \quad \forall q \in Q \text{ and } \forall b \in B_q \quad (25)$$

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.,

$$\kappa = \sum_{qb} w_{qb}(\rho, \kappa) x_{qb} \quad (26)$$

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.

[10] 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., $\overline{PrConv'_q}(\rho, \kappa) = \eta\left(\overline{PrConv_q}(\rho, \kappa), MSB\right)$.

Exhaustive Greedy PMCKP.

For any given ρ (0, for now) and κ, values and weights and 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 clearly 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.

Pseudocode that implements a "dynamic" version of the GreedyMCKP algorithm appears in Algorithm 2. This algorithm 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.

Algorithm 2 DynamicPMCKP

Input: $\forall q, \vec{v}_q, \vec{w}_q, C$
Output: $\forall q, \vec{x}_q$
  $\kappa = 0$ {Capacity used}
  $\vec{y}_q = \vec{0}$ {Incremental items taken}
  $\forall q, i_q = 0$ {Incremental item just taken}
  repeat
    for $s \in S$ **do**
      $n_q :=$ `NextUndominatedItem`$(\vec{v}_q, \vec{w}_q, i_q, \kappa, C)$
      $w :=$ `IncrementalAdjustedWeight`$(\vec{w}_q, i_q, n_q, \kappa, C)$
      $v :=$ `IncrementalAdjustedValue`$(\vec{v}_q, i_q, n_q, \kappa, C)$
      $e_q := v/w$ {incremental adjusted efficiencies}
    end for
    $s^* := \arg\max_q e_q$ {s^* is an item of maximal efficiency}
    $e^* := \max_q e_q$ {e^* is the maximum efficiency}
    if $e^* > 0$ **then**
      take item s^*: i.e., $y_{s^* n_{s^*}} := 1$
      update κ
      $i_{s^*} := n_{s^*}$
    end if
  until $e^* \leq 0$
  $\forall q, \vec{x}_q =$ `ConvertIncrementalItemsTakenToItems`$(\vec{y}_q)$
  return $\vec{x}_q$

Hybrid Greedy PMCKP.

Eliminating dominated items requires sorting, meaning there is an $O(n \log n)$ operation inside the main loop of DynamicMCKP. HybridMCKP 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. DynamicMCKP is $O(n^2 \log n)$, where n is the total number of items across item sets, while HybridMCKP is only $O(n^2)$ [3].

5.2 Cross-Day Dependencies

We are now ready to present the multi-day TAC AA bidding problem,[11] the most complex problem studied here:

$$\max_x \sum_{d=t}^{D} \sum_{q \in Q} \sum_{b \in B_q} v_{dqb}(\kappa_d, \rho_d) x_{dqb} \tag{27}$$

$$\sum_{b \in B_q} x_{dqb} \leq 1 \quad \forall q \in Q, \ t \leq d \leq D \tag{28}$$

$$x_{dqb} \in \{0, 1\} \quad \forall \ q \in Q, \ b \in B_q, \ t \leq d \leq D \tag{29}$$

Here, t is the current day; D is the total number of days; W is the window (i.e., number of days) in which unused capacity can be accumulated; and c_d is the (known) number of conversions on day d, for $d < t$.

Eqs. (28) and (29) are the same as in the single-day problem, except that they have an additional day index, since we can now choose an amount to bid on each future day. Eq. (27) is the same as Eq. (23), the single day objective function, except that we are now trying to maximize value over all future days, not just the current day. The values on day d depend on how many conversions are made on that day (κ_d) and on how many have already been made on previous days inside the capacity window (ρ_d):

$$\kappa_d = \sum_{qb} w_{dqb}(\kappa_d, \rho_d) x_{dqb} \tag{30}$$

$$\rho_d = \begin{cases} c_d & \text{if } d < t \\ \sum_{i=d-W}^{d-1} \kappa_i & \text{otherwise} \end{cases} \tag{31}$$

When $d < t$, the value of ρ_d is known, but when $d \geq t$, the value of ρ_d must be calculated based on κ_d.

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.

6. EVALUATION

[11]This is an expected value formulation of the bidding problem, not a fully stochastic formulation, because we use *expected* conversion probability.

We now evaluate the bidding heuristics we presented in the previous sections. We first present our controlled testing framework in more detail, then describe our experimental design, and finally present our results.

6.1 Testing Framework

We refer to our controlled testing framework as the *optimization challenge*. At a high level, the optimization challenge (1) removes the game-theoretic complexities of the TAC AA game and (2) optionally controls the accuracy of an optimization algorithm's models. The optimization challenge takes as input a previously played TAC AA game[12]. One of the original agents A is removed from the game, and the optimization algorithm we are evaluating is provided with the daily information that A would normally have received. The optimization algorithm requires some models to make its decisions; these models can either be actual models that make predictions based on the received daily information, or they can be artificially improved models that are created by simulating the results of the agent placing various bids. When the algorithm sends its daily decisions to the optimization challenge, that day is re-simulated with these decisions plus the old decisions made by the other agents still in the game.

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 agent decisions. In general, we believe the game-theoretic aspect of a competition like TAC AA does not have much effect on the design of agents' strategies. Finding equilibria theoretically is intractable, and designing a strategy to affect the other players' actions in a certain way is also difficult, both because of the noise associated with agents' inputs (e.g., from the behavior of search engine users) and because the other agents are not necessarily rational players. However, we run tests to verify that the optimization challenge results are representative of the TAC AA game.

6.2 Experimental Design

We run tests with the agents from the stylized problem (GreedyMCKP, EquateROI, and EquatePM), as well as agents from the more complex problem (ExhaustiveMCKP, DynamicMCKP, and HybridMCKP). Using the TAC AA game server and the optimization challenge described above, we perform the following tests:

1. We evaluate performance in actual TAC AA games. A subset of agents play N games against the most competitive set of agents available from the TAC agent repository[13].

2. As an intermediate step, we verify that the optimization challenge, despite removing some game-theoretic aspects of the game, is still representative of the full TAC AA game. We run each agent in the optimization challenge on N finals games from the 2009 TAC AA tournament. We would like to see that the profit ordering of our agents is similar in both actual TAC AA games and in the optimization challenge games.

[12]Logs of competition games are publicly available at http://aa.tradingagents.org/servers/

[13]Repository agents are located at `http://www.sics.se/tac/showagents.php`.

Agent	Set 1	Set 2	Set 3
TacTex	80.76	79.86	81.84
DynamicMCKP	**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)

3. Again using the optimization challenge, we run the GreedyMCKP algorithm and rule-based algorithms with various fixed daily target capacities, and compare these results to performance using the ExhaustiveMCKP, DynamicMCKP, and HybridMCKP. From these results, we quantify the benefit of dynamically setting target capacity each day as opposed to targeting a fixed daily amount of target capacity.

4. Finally, we evaluate optimization algorithm performance under varying degrees of model accuracy. For this paper, we focus on the $PrClick_q$ and CPC_q models. Using the optimization challenge, we run an MCKP and a rule-based agent on the 2009 TAC AA finals games, evaluating how performance changes when a single model becomes less accurate.

6.3 Results

TAC AA Games.

Our results from running 60 TAC AA games are shown in Table 1. From these games, we observe that DynamicMCKP and EquateROI are both finishing in 2nd place against the highest- performing agents in the TAC agent repository. To understand how similarly these agents are bidding, we will analyze more detailed statistics of these agents in the optimization challenge. Also of note is that the profit ordering of opponents is generally consistent across each set of games. Of the seven opponents in the three sets, there were only two inconsistent permutations to the ordering of opponents (munsey and epflagent in Set 1, and QuakTAC and MetroClick in Set 3). This provides some initial evidence that agents are not strongly affected by a single agent changing strategies.

Optimization Challenge on Finals Games.

Table 2 presents the detailed results from running each of our agents in 40 games from the 2009 TAC AA finals. We first note that the agents we ran in the real TAC AA games have maintained their profit ordering across both TAC AA games and the optimization challenge. There were two possible reasons for which profit could have changed across these frameworks: (1) because this second group of tests was in the optimization challenge, opponents were not able to respond to our agents' actions, and (2) because not all finals agents were available in the TAC AA repository, these two groups of experiments consisted of slightly different sets of

Agent	Profit	Avg Pos	CPC	ClickPr	ConvPr	Sales in CS	Sales in MS	Overcap
DynamicMCKP	**75.93**	2.49	0.46	0.21	0.17	0.41	0.85	1.38
HybridMCKP	75.85	2.42	0.48	0.22	0.17	0.40	0.86	1.40
ExhaustiveMCKP	75.01	2.70	0.42	0.20	0.14	0.41	0.78	1.45
EquateROI	**73.47**	2.33	0.48	0.20	0.14	0.38	0.86	1.40
EquatePM	**64.70**	3.06	0.34	0.13	0.11	0.44	0.38	1.53
EquateCPOD	65.88	3.18	0.33	0.12	0.12	0.44	0.37	1.51
EquatePPCD	65.72	3.07	0.33	0.13	0.11	0.44	0.37	1.55

Table 2: Average Performance (40 simulations of the finals games)

opponents. The fact that profit orderings were consistent suggests that the optimization challenge, while making simplifications to the full TAC AA game, still represents the aspects of TAC AA that most influence agent profit.

Comparing the rule-based algorithms, we note that, as expected, EquatePM, EquateCPOD, and EquatePPCD perform almost identically, both in terms of final profit and more detailed behavior. This verifies our claim that equating each of these is equivalent.

EquateROI, on the other hand, has clearly different behavior from the other rule-based algorithms, and is also the most successful of these algorithms. We note that that vast majority of EquateROI's sales are in its manufacturer specialty (MS). This behavior is easily explained by examining the EquateROI algorithm. Recall from Section 2 that all products in an agent's manufacturer specialty have an increased unit sales price compared to other products. Because the agent's manufacturer specialty boosts the USP_q for some queries, by Algorithm 1, the target CPC for those queries will be higher. If the agent is still able to reach its target sales level, target ROI increases, driving CPC values down across all queries. Eventually CPCs become so low for queries without the manufacturer specialty bonus that these queries have bids below the minimum reserve score. Because EquateROI can comfortably sell in only its manufacturer specialty and still place as one of the top agents, our conclusion is that perhaps the manufacturer specialty bonus in TAC AA is too high, since it is encouraging an almost complete focus in specific queries.

Target Capacity Experiments.

Figure 1 provides results from our fixed capacity experiments. Focusing first on the MCKP algorithms, note that HybridMCKP, DynamicMCKP, and ExhaustiveMCKP all perform better than GreedyMCKP at any fixed capacity. This quantifies the cost of choosing a fixed target capacity across days, and also indicates that all of the other MCKP algorithms besides GreedyMCKP must be varying their target capacity throughout the game, or else their average profit would be no higher than that of GreedyMCKP. EquateROI is also choosing a fixed target capacity throughout the game, and is performing very similarly to GreedyMCKP. This indicates that the true benefit of our MCKP algorithms, given our current levels of model accuracy, comes from their ability to target a specific capacity for each day. The rule-based algorithms could perhaps be improved by introducing a new variant that adjusts not just target ROI (or PM, etc.), but also target capacity. We leave the design of this algorithm for future work.

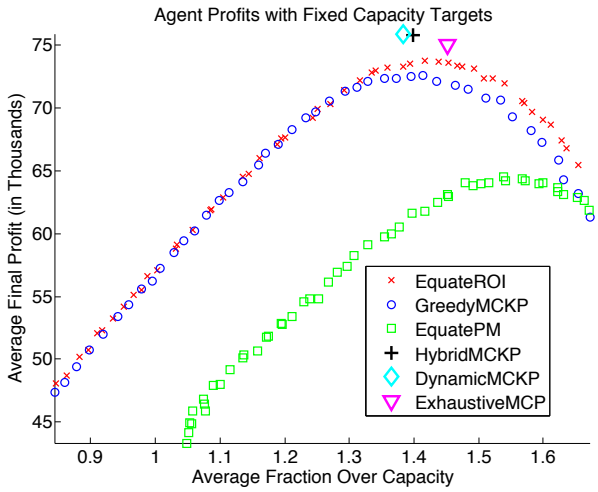

Figure 1: Profits at various fixed target capacities.

Model Accuracy Experiments.

We now use the optimization challenge to evaluate performance of algorithms with different levels of model accuracy (Figure 2). We run the DynamicMCKP and EquateROI algorithms using their standard models, except that varying levels of Guassian noise are added to one of the models. Because the rule-based algorithm does not use the *PrClick* or *CPC* models, its profits are constant across any of these levels of model accuracy. In these experiments DynamicMCKP performed better than EquateROI with our actual models but became worse as we added more noise. Also, DynamicMCKP is more robust to noise in its *CPC* model than its *PrClick* model. This suggests that improving our *PrClick* model is a more valuable focus of effort than improving the *CPC* model, if we believe that trends in performance vs. model accuracy continue as model accuracy improves beyond its current level. 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.

7. CONCLUSION

In this paper, we have presented a suite of known bidding algorithms to solve a stylized ad auctions problem and have extended these algorithms to the Trading Agent Competition for Ad Auctions (TAC AA) domain. We have designed a framework to isolate the optimization component of a bid-

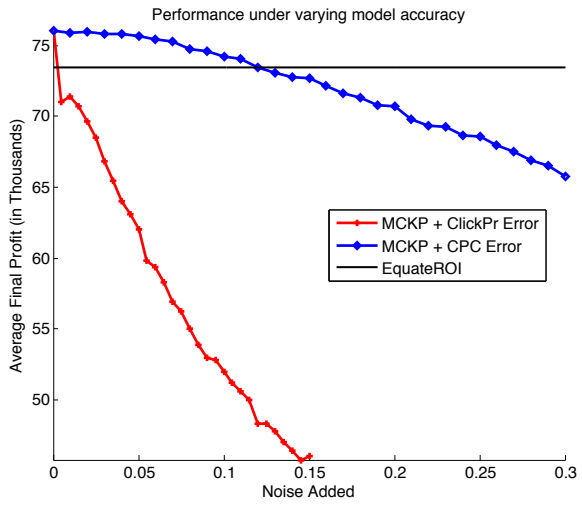

Figure 2: Agent profits with noise added to the click probability and CPC models.

ding agent, and shown that performance in this framework is representative of performance in actual TAC AA games. We evaluate each optimization algorithm's performance with respect to varying degrees of model accuracy, and find that a simple rule-based algorithm can successfully compete in this domain, although its relative performance will likely decrease as more sophisticated models are developed. We quantify the benefits of dynamically choosing a sales target each day, and show that there is no significant difference between using our dynamic greedy algorithm and performing an exhaustive search over target capacities.

In continuing this work, our goal is to develop algorithms that can handle additional dependencies that are relevant to real-world advertisers, such as dependencies across phrase-match queries and reputation/branding effects that may result in increased bids having a positive effect on other queries.

7.1 ACKNOWLEDGMENTS

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.

8. 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 knapsack-based approach to bidding in ad auctions. In *In the 19th European Conference on Artificial Intelligence (ECAI)*, 2010.

[4] 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.

[5] 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.

[6] 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.

[7] J. Feldman, S. Muthukrishnan, M. Pal, and C. Stein. Budget optimization in search-based advertising auctions. In *ACM EC*, pages 40–49, 2007.

[8] A. Greenwald, V. Naroditskiy, and S. Lee. Bidding heuristics for simultaneous auctions: Lessons from tac travel. In *Workshop on Trading Agent Design and Analysis*, July 2008.

[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] D. Kempe and M. Mahdian. A cascade model for externalities in sponsored search. In *WINE '08: Proceedings of the 4th International Workshop on Internet and Network Economics*, pages 585–596, Berlin, Heidelberg, 2008. Springer-Verlag.

[14] B. Kitts and B. J. LeBlanc. Optimal bidding on keyword auctions. *Electronic Markets*, 14(3):186–201, 2004.

[15] S. Lahaie. An analysis of alternative slot auction designs for sponsored search. In *EC '06: Proceedings of the 7th ACM conference on Electronic commerce*, pages 218–227, New York, NY, USA, 2006. ACM.

[16] A. Mehta, A. Saberi, U. V. Vazirani, and V. V. Vazirani. Adwords and generalized on-line matching. In *Proc. FOCS*, pages 264–273, 2005.

[17] S. Muthukrishnan, M. Pál, and Z. Svitkina. Stochastic models for budget optimization in search-based advertising. In *WINE*, pages 131–142, 2007.

[18] 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.

[19] 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)*, 2010.

[20] D. Pardoe and P. Stone. The 2007 tac scm prediction challenge. In *AAAI 2008 Workshop on Trading Agent Design and Analysis*, 2008.

[21] H. R. Varian. Position auctions. *International Journal of Industrial Organization*, 25(6):1163–1178, December 2007.

[22] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[23] 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.

[24] 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.