To appear in *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems* (AAMAS 10), Toronto, Canada, May 2010.

# TacTex09: A Champion Bidding Agent for Ad Auctions

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

## ABSTRACT

In the Trading Agent Competition Ad Auctions Game, agents compete to sell products by bidding to have their ads shown in a search engine's sponsored search results. We report on the winning agent from the first (2009) competition, TacTex. TacTex operates by estimating the full game state from limited information, using these estimates to make predictions, and then optimizing its actions (daily bids, ads, and spending limits) with respect to these predictions. We present a full description of TacTex along with analysis of its performance in both the competition and controlled experiments.

## Categories and Subject Descriptors

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

## General Terms

Algorithms, Experimentation, Economics

## Keywords

trading agents, sponsored search, ad auctions

## 1. INTRODUCTION

Sponsored search [4] 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* [5] 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).

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. As a result, much effort has gone into developing automated strategies that can bid intelligently. A number of companies offer software for such a purpose, and the problem has also attracted a growing number of researchers. One barrier to effective research is that it can be difficult to benchmark automated strategies in a live business environment, both due to the proprietary nature of the systems and due to the high cost of errors. Research in simulation, on the other hand, suffers from the need to realistically model other bidders and the difficulty of comparing results from different simulated environments. The Trading Agent Competition Ad Auctions Game (TAC/AA) aims to address these issues by providing a competitive environment in which independently created agents can be tested against each other over the course of many simulations.

In this paper, we report on TacTex, the winning agent in the first TAC/AA competition. TacTex operates by estimating the full game state from limited information, using these estimates to make predictions, and then optimizing its actions with respect to these predictions. After summarizing the TAC/AA rules, we describe the full TacTex agent, and then we present an analysis of its performance in both the competition and controlled experiments. In this paper we have chosen to offer limited details on some agent components and emphasize the integration of these components into a complete, practical system. A more detailed agent description is available as a technical report [6].

**Cite as:** TacTex09: A Champion Bidding Agent for Ad Auctions, David Pardoe, Doran Chakraborty, and Peter Stone, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.


## 2. GAME DESCRIPTION

In this section, we provide a summary of those parts of the TAC/AA game that are most important for understanding the design of TacTex. For full details, see the game specification [2].

**Overview:** In each TAC/AA game, eight autonomous 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 (*flat*, *pg*, and *lioneer*) and by component (*tv*, *dvd*, and *audio*) 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 *null*, i.e. missing. There are thus 16 total query types, divided into three *focus levels*: F0 (the query with both manufacturer and component null), F1 (the six queries with one null and one specified), and F2 (the nine queries with both specified). 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, ii) an ad, and iii) a spending limit (optional). Ads can be either *targeted* (specifying both a manufacturer and prod-

| | *Agent actions* | | | | *Results* | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| *Advertiser* | *Bid* | *Sq. bid* | *Ad* | *Sp. limit* | *CPC* | *Imps* | *Clicks* | *Convs* | *Impression range* | *Avg pos* |
| MetroClick | 0.315 | 0.109 | generic | 50.93 | 0.310 | 426 | 164 | 16 | | 1.000 |
| QuakTAC | 0.266 | 0.107 | lioneer:dvd | - | 0.194 | 718 | 156 | 6 | | 1.593 |
| TacTex | 0.235 | 0.091 | generic | 0.236 | 0.201 | 77 | 1 | 0 | | 3.000 |
| UMTac09 | 0.216 | 0.078 | generic | 7.583 | 0.209 | 700 | 36 | 6 | | 2.719 |
| munsey | 0.190 | 0.075 | generic | - | 0.174 | 718 | 16 | 2 | | 3.675 |
| epflagent | 0.214 | 0.068 | generic | - | 0.184 | 641 | 3 | 0 | | 4.510 |
| AstonTAC | 0.158 | 0.059 | generic | 500.0 | 0.133 | 292 | 1 | 0 | | 4.938 |
| Schlemazl | 0.062 | 0.020 | flat:dvd | 5.617 | - | 0 | 0 | 0 | | - |

**Table 1: Results for the query *null:dvd* from one game day of the 2009 TAC/AA finals**

uct) or *generic* (specifying neither). (The set of 16 (bid, ad, spending limit) tuples can be said to be an advertiser's action space, and sections 3-8 essentially describe how TacTex maps its observations into this space each day.) 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.

**Users:** There is a fixed pool of users, each of which remains interested in a specific product throughout the game and only submits queries corresponding to this product. However, users cycle through states corresponding to the focus levels according to a specified transition model. Users begin in a non-searching (NS) state, and can then transition through a searching (IS) state (which may submit a query of any focus level but will not make a purchase) to one of three buying states (F0, F1, or F2, each of which submits a query of the corresponding focus level and makes a purchase with a probability that increases with the focus), and eventually back to the non-searching state. For each product, the total number of users in any state can vary widely and rapidly.

**Click model:** Every searching or buying user submits one query per day and then proceeds through the resulting ads in order of advertiser ranking. When an advertiser's ad is shown, it is said to receive an *impression*, but not all impressions result in clicks. The default user behavior is as follows. If a user submitting query $q$ reaches the ad of advertiser $a$, the probability of a click is $e_q^a$, a hidden parameter drawn randomly at the start of each game. If the user clicks, it will then *convert* (make a purchase) with a probability dependent on the user's focus level. For each conversion, the advertiser receives $10. (This amount is technically the sales profit before considering advertising costs, but we will simply refer to it as the agent's *revenue*). If the user does not convert, it proceeds to the next ad with probability $\gamma_q$, another randomly drawn, hidden game parameter. Thus, the higher the position of the ad, the more likely it is to be clicked. A number of factors can modify this default behavior. First, if an advertiser's ad is targeted, the click probability is raised or lowered depending on whether the ad matches the product desired by the user. Second, each advertiser has a component and manufacturer specialty. If the product desired by the user matches the component specialty, the conversion probability is increased, and if it matches the manufacturer specialty, the advertiser's revenue is increased. Finally, the conversion probability decreases if the advertiser has exceeded its capacity, as described below.

**Auctions:** Ads are ranked using a generalized second price auction. Rather than ranking ads solely by bids, the search engine also considers click probability. If for query type $q$ an advertiser's bid is $b_q$ and its default click probability is $e_q^a$, then its *squashed bid* is defined as $(e_q^a)^\chi b_q$, where $\chi$ is a random but known game parameter. Ads are ranked by squashed bid, and each time an advertiser's ad is clicked, it pays the minimum amount it could have bid while still beating the squashed bid of the advertiser ranked below it.

**Capacity:** Each advertiser is assigned a capacity $c$ which serves as a soft constraint on how many products it can sell (of any type) over a five day period. Whenever an advertiser's ad is clicked, if the number of products $n$ sold over five days (including those sold so far on the current day) exceeds $c$, then the conversion probability is multiplied by a *distribution constraint* equal to $0.995^{n-c}$. Note that the distribution constraint changes during the day as the advertiser sells more products.

**Information:** Advertisers must operate in the face of limited information about customers and competitors. For each query type, the advertiser receives a daily report stating how many impressions, clicks, and conversions the advertiser received and the average cost per click (CPC). The only other information available is a report on the ad used by each advertiser and the average position of that ad. An advertiser that wishes to increase its number of clicks would therefore have little information about how much it would cost to increase the position of its ad or how many clicks it might expect in the new position. Advertisers are also unaware of the types (specialties and capacities) of other advertisers.

**Example:** Table 1 shows the results for the query *null:dvd* from a sample game day. The eight advertisers are shown in order of their squashed bids, which differs from the order of the true bids due to differing $e_q^a$ values. The ads and spending limits (where used) of each agent are also shown. The results of these actions are shown on the right side of the table: the cost per click, impressions, clicks, and conversions. In addition, the *impression range* column shows a graphical representation of the period for which each advertiser's ad was shown, with the day progressing from left to right. On this day, 718 users submitted the query *null:dvd*, but due to spending limits, only two agents, QuakTAC and munsey, received the full 718 impressions. TacTex was the first agent to hit its spending limit (after a single click - this was a probe, as described later). At that point, all lower advertisers increased by one position, and since epflagent reached the fifth position, its ad began to be shown. Hence, the impression range column shows epflagent starting where TacTex stopped. Although Schlemazl reaches the fifth position at the end of the day, its ad is not shown because its bid is below the reserve. Finally, the average position for each advertiser is shown. Note that the average positions are not in the same order as the squashed bids, and that the average is only for the period in which the ad was shown (thus never above 5). From this table, the only information available to TacTex was its own row (except for the squashed bid) and the ad and average position columns. Much of Tac-

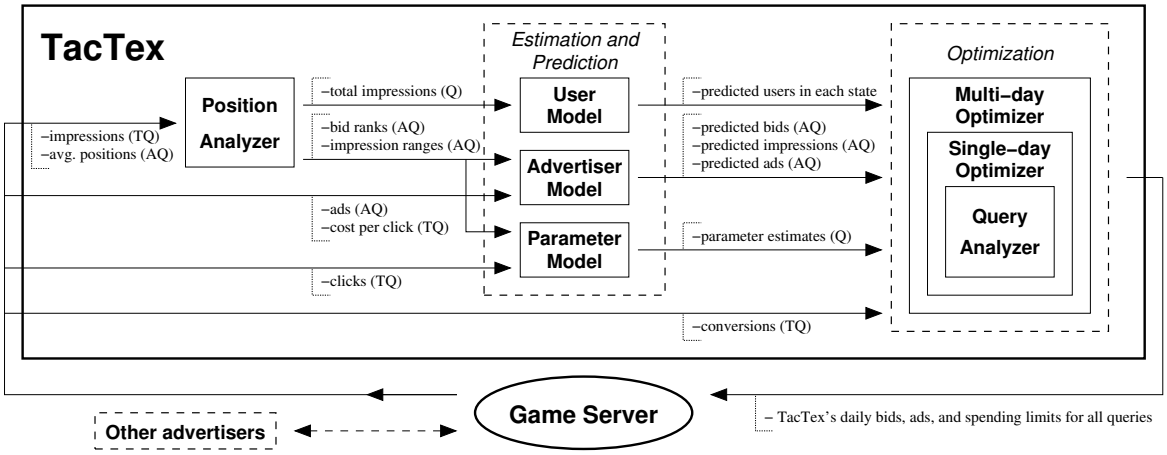


**Figure 1: Flow of information in TacTex. T = TacTex only, A = all advertisers, Q = all queries.**

Tex's computational effort is devoted to estimating the rest of this information so that its decisions can be based on as much information as possible.

# 3. TACTEX OVERVIEW

At a high level, TacTex operates by making predictions or estimates concerning various factors (such as unknown game parameters, user populations, and competitor bids) and then by finding the optimal actions given this information. These tasks are divided among a number of modules that we describe in detail in the following sections. Here, we give an overview of these modules. Figure 1 depicts the relationship between the modules, including the inputs and outputs of each.

At the start of each new day, the game server sends TacTex a report on the results of the previous day. The first module to be called is the Position Analyzer, a preprocessor that converts some of this information into a more useful format. The goal of the Position Analyzer is essentially to reconstruct the impression range column of Table 1 for each query type.

TacTex then performs all necessary prediction and estimation using three modules. The User Model uses the total number of queries for each query type to estimate the composition of each of the nine user populations. From these estimates, predictions about future user populations can be made. The Advertiser Model takes information relating to the actions of other advertisers and predicts the actions these advertisers will take in the future. The Parameter Model maintains estimates of unknown game parameters by finding those parameters that best fit the known auction outcomes.

Finally, TacTex must use these predictions and estimates to choose the optimal bids, ads, and spending limits to submit to the game server for the next day. The Query Analyzer uses the information received to compute the expected outcomes of actions, such as how many clicks and conversions would occur for a given query type. If there were no distribution constraint, then TacTex could optimize for each query type independently, but instead it must choose how to allocate its available capacity among query types and even among multiple days. The Multi-day Optimizer is responsible for dividing capacity among the remainder of the game days, and it calls the Single-day Optimizer to divide each day's capacity among query types using the information provided by the Query Analyzer.

# 4. POSITION ANALYZER

For each query type, the Position Analyzer takes the average advertiser positions and attempts to extract i) the ranking of the squashed bids, and ii) the first and last impression for each advertiser. Each advertiser's average position can be expressed as a function of the bid rankings and first and last impressions of all other advertisers, and while this system of equations cannot be solved directly, it is possible to efficiently search the space of unknown values to find a correct or nearly correct solution. We leave the details of the search algorithm to the technical report.

# 5. USER MODEL

The User Model maintains estimates of the user population states by using a particle filter for each of the nine populations. Each of the 1000 particles used per filter represents a distribution of the population's 10,000 users among the user states (NS, IS, F0, F1, and F2). At the start of the game, the initial particles reflect the possible populations resulting from the game server's initialization process. Each succeeding day, a new set of particles is generated from the old. For each new particle to be generated, an old particle is selected at random based on weight, and the new particle's user distribution is randomly generated from the old particle based on the user transition dynamics.

Although many observations depend on the user populations, the most informative are the total impressions for each of the nine F2 queries. Recall that all F2 users submit an F2 query, and each IS user has a 1/3 probability of doing so. The number of F2 impressions resulting from a given user population thus follows a binomial distribution, and the User Model uses this fact to weight each particle according to its relative likelihood.

The resulting set of particles represents the estimated probability distribution over the user population state on the previous day. To obtain the expected user population $n$ days in the future, the User Model updates each particle $n+1$ times according to the user transition dynamics and takes the weighted average of the particles.

# 6. ADVERTISER MODEL

The Advertiser Model makes three types of predictions about the actions of the competing advertisers.

## 6.1 Impression predictions

For each query type, the Advertiser Model predicts the

maximum number of impressions that each advertiser could receive before hitting its spending limit. In cases in which an advertiser did not hit its spending limit by the previous day's final impression, the Advertiser Model assumes that the advertiser effectively had no spending limit and will also not hit its spending limit on the coming day. Otherwise, the prediction is set to the number of impressions received by the advertiser on the previous day.

## 6.2 Ad predictions

The Advertiser Model also predicts the ads (targeted or generic) that other advertisers will choose. For each query type, the Advertiser Model maintains a count for all the ads it has seen so far from each advertiser. The predicted ad for that query is then the majority ad, i.e., the ad having the highest count amongst all posted ads for that query type.

## 6.3 Advertiser bid estimation

The third task performed by the Advertiser Model is to maintain estimates of the bids submitted by each advertiser for each query and then to predict what future bids will be. Advertiser bid estimation is a hard problem because the bidding dynamics of other advertisers are unknown – while bids often change only gradually, it is not uncommon for large jumps to occur. In addition, the Advertiser Model receives only partial information about the bids of other advertisers (the bid ranks and TacTex's CPC). During the development of TacTex, we created two very different and independently designed bid estimators. Our preliminary testing did not show either approach to be superior; however, we found that an ensemble approach that averaged the output of the two estimators outperformed either one alone, and so the Advertiser Model uses both estimators in this fashion. Here we present both bid estimators in limited detail. The primary difference between the two is that the first estimator models all advertisers' bids jointly, while the second models bids independently.

**First bid estimator :** The first approach uses particle filtering to estimate the bids of other advertisers. We use one particle filter for each of the 16 query types. Each of the 1000 particles per filter represents one set of bids for all other advertisers for that query type. Associated with the particles is a probability distribution that gives the likelihood of each particle representing the current state. On each new day, each particle filter samples from the underlying distribution to obtain the next set of particles. Then it updates each particle based on the observations received that day, i.e., the cost per click and bid rankings. Once the particles have been updated, the filter recomputes the probability distribution for the new set of particles.

The sampling step is straightforward. The next step in particle filtering is to update each particle using the known dynamics. In the absence of such known dynamics, we propose a departure from the traditional vanilla particle filter and use a part of the observation to do the update. Let $cpc_{t+1}$ and $r_{t+1}$ be the cost per click and ranking for that query seen on day $t+1$. We use these values to reset some of the bids made by other advertisers in some particles (where necessary) in an attempt to improve the respective particle. On each iteration of the update, the bid, $b^x_{t+1}$, of an advertiser $x$ is adjusted while holding the bids of the other advertisers fixed. The bids are adjusted for only those cases where the order of a bid is incorrect with respect to the known bid ranking. The two cases where the order is correct and there is no need for bid adjustment are:

$$\begin{aligned} r^x_{t+1} > (r^{TacTex}_{t+1} + 1) \quad \wedge \quad b^x_{t+1} < cpc_{t+1} \\ r^x_{t+1} < r^{TacTex}_{t+1} \quad \wedge \quad b^x_{t+1} > b^{TacTex}_{t+1} \end{aligned} \tag{1}$$

The conditions when $b^x_{t+1}$ needs to be updated, and how these updates are made, are mentioned below. $rand(a, b)$ denotes a random draw from the range $(a, b)$. $z$ denotes the particle and $z(r)$ denotes the bid of the advertiser ranked $r$ in $z$.

$$b^x_{t+1} = \begin{cases} cpc_{t+1} & \text{if } r^x_{t+1} = r^{TacTex}_{t+1}+1 \\ rand(0, \text{least bid value in } z) & \text{if } r^x_{t+1} = \text{undefined} \\ rand(z(r^x_{t+1}+1), z(r^x_{t+1}\text{-}1)) & \text{otherwise} \end{cases}$$

Note that the "otherwise" case excludes the conditions mentioned in 1. The whole process is repeated a fixed number of times, holding one advertiser fixed each time, with each iteration improving upon the former (20 iterations is sufficient). At the end of this update step, we have a better particle having closer predictions of other advertiser bids.

Next comes the step of recomputing the probability distribution of the sampled particles. Although the true likelihood of a particle whose ranking does not match the true ranking $r_t$ is zero, there may be few particles with the correct ranking, and so we instead use a likelihood function designed to give some weight to all particles. We compute the difference of ranking for each advertiser from the two available sources, i.e., $r_t$ and the rank from $z$. For a distance $\delta$, we define $\kappa(\delta) = exp(-\frac{\delta^2}{4.9})$. The likelihood of each particle is set to the product of these $\kappa(\delta)$ values over all advertisers, and thus the particles whose predicted rankings are closer to $r_t$ get assigned higher values. These values are normalized over all 1000 particles to give the true probability distribution captured by the particles.

**Second bid estimator:** The second approach tries to compute each advertiser's bid separately. The bids are represented as discretized values rather than a changing set of particles. The bid space $[0, 3.75]$ is discretized into values $v_1$ through $v_{100}$ by setting $v_i = 2^{i/25-2} - 0.25$ (thus $v_{50} = 0.75$). Discretizing the bid space in this way allows better coverage of low bids, which are most common, while still maintaining the ability to represent very high bids.

On day $t + 1$, for each advertiser $x$, we wish to estimate the distribution of the new bid, $b^x_{t+1}$, over these discrete $v$ values, conditional on the observed ranking $r_{t+1}$, previous bids $b^x_1$ ... $b^x_t$, and the bids of other advertisers, $B^{-x}_{t+1}$. We make the simplifying assumptions that $r_{t+1}$ and $b^x_1$ ... $b^x_t$ are conditionally independent given $b^x_{t+1}$, and that $B^{-x}_{t+1}$ and $b^x_1$ ... $b^x_{t+1}$ are independent. Applying Bayes' rule twice and rearranging, we derive:

$$\begin{aligned} Pr(b^x_{t+1} = v_i | r_{t+1}, B^{-x}_{t+1}, b^x_1 ... b^x_t) \propto \\ Pr(r_{t+1} | B^{-x}_{t+1}, b^x_{t+1} = v_i) Pr(b^x_{t+1} = v_i | b^x_1 ... b^x_t) \end{aligned} \tag{2}$$

The first term in the R.H.S of Equation 2 is the probability of the observation while the second term is the transition model of bids for $x$, both unknown.

We model bid transitions by assuming that bids change in one of 3 ways. First, with 0.1 probability, $b^x_{t+1}$ jumps uniformly randomly to one of the $v_i$ values. This case covers sudden jumps that are difficult to model. Next, with 0.5 probability, $b^x_{t+1}$ changes only slightly from $b^x_t$. We assume that the probability of changing from $v_i$ to $v_j$ is proportional to $\phi_{0,6}(|i - j|)$, where $\phi_{0,6}$ is the density function of the zero-mean normal distribution with variance 6. Finally,

we assume that with 0.4 probability, the bid changes according to a similar distribution, but the change is with respect to the bid 5 days ago, $b^x_{t-4}$. This case captures the fact that bids often follow 5 day cycles due to the 5 day capacity window. Let $tr(j,i)$ denote the the resulting probability of the bid transitioning to $v_i$ from $v_j$ using the above normal distribution and normalizing. Then, summing the three cases gives us the following: $Pr(b^x_{t+1} = v_i | b^x_t = v_j, b^x_{t-4} = v_k) = 0.001 + 0.5tr(j,i) + 0.4tr(k,i)$. Our estimate for $b^x_1$ is initialized to a distribution consistent with observed game data, and when $t < 5$, we substitute $b^x_1$ for $b^x_{t-4}$. The probabilities for the three cases were chosen to provide robustness to a variety of agent behaviors in pre-competition experiments.

The observation probabilities are now conditioned on a single advertiser's bid, rather than a set of bids as in the first bid filter. Let $y$ be another advertiser in the game apart from $x$. If advertiser $y$ is TacTex, then we know the bid; otherwise we have a distribution representing our estimate of the bid for $y$. Thus the conditional probability of the set of rankings $r_{t+1}$ given a fixed $b^x_{t+1} = v_i$ and a fixed value of the distribution $B^{-x}_{t+1}$ is :

$$P = \Pi_{\forall y \neq x} \begin{cases} Pr(b^y_{t+1} > v_i) & \text{if } r^x_{t+1} > r^y_{t+1}, \\ Pr(b^y_{t+1} < v_i) & \text{if } r^x_{t+1} < r^y_{t+1}, \\ 1 & \text{otherwise} \end{cases} \quad (3)$$

where $P$ denotes $Pr(r_{t+1} | b^x_{t+1} = v_i, B^{-x}_{t+1})$. Note that ranks will only be equal if neither advertiser had any impressions; in this case we have no information about the relative bids. Also, whenever $y$ is TacTex, the R.H.S will be 1 or 0 since we know our own bid. Finally, we have been treating $B^{-x}_{t+1}$ as if it were known, but in fact these are the other advertisers' bids that we are trying to estimate simultaneously. We address this problem by applying Equation 2 for 10 iterations, using the latest estimates for each bid distribution, as this resulted in sufficient convergence in testing.

# 7. PARAMETER MODEL

Recall that for each query type $q$, the parameter $\gamma_q$ represents the probability that a user will progress from one ad to the next, while each advertiser $a$ has a parameter $e^a_q$ that affects the probability of a user clicking its ad. Given the bid rankings and impression ranges computed by the Position Analyzer and the User Model's population estimate, we can determine the distribution over the number of clicks that TacTex would receive for any set of these parameter values. The Parameter Model maintains a joint distribution over $(\gamma_q, e^{TacTex}_q)$ pairs by discretizing the possible space of values uniformly and setting the likelihood of each pair to be proportional to the product of the probabilities of each day's number of clicks. There is insufficient information to effectively estimate $e^a_q$ values of other advertisers, and so we assume they equal the mean possible value.

# 8. OPTIMIZATION

To this point, we have described those modules that estimate the game state and make predictions about the future. We now turn to the challenge of using this information to select actions. In particular, each day TacTex must choose bids, ads, and spending limits for each query. The key factor in the optimization process is the distribution constraint. Recall that while there is no hard cap on capacity, exceeding a certain number of conversions results in a reduced conversion rate. Beyond some point, marginal returns per conversion can become negative. As a result, TacTex performs optimization by reasoning about conversions and then choosing actions expected to result in those conversions, rather than reasoning directly in the space of possible actions.

The optimization process consists of three levels: a Multi-day Optimizer (MDO), a Single-day Optimizer (SDO), and a Query Analyzer (QA). Because we want to maximize profit for the entire game, not a single game day, the top-level decision that must be made is how many conversions to target on each remaining game day, and this decision is made by the MDO. Computing the expected profit for a given day and conversion target requires deciding how to divide the conversions among the 16 query types, and the MDO calls the SDO to perform this task. Finally, the SDO calls the QA to i) determine the bid, ad, and spending limit that are expected to result in a given number of conversions, and ii) compute the expected cost and revenue from those conversions. We describe these three levels from the bottom up.

## 8.1 Query Analyzer

For any given bid, ad, and spending limit, it is fairly straightforward to determine the expected cost, revenue, and conversions from a specific query type. The QA does this by taking the expected user population, iterating through all impressions, computing our position and CPC, and then computing the probability that the user i) reaches our ad, ii) clicks on it, and iii) converts. (Note that at this level we are not considering the distribution constraint, which may lower the conversion rate.) However, the problem we face is essentially the reverse: the QA is given a conversion target and needs to determine the bid, ad, and spending limit that will produce those conversions in the most profitable way. Up to a certain point, raising either the bid or the spending limit will increase the number of conversions, while the effect of ad choice depends on the user population, so there may be a number of ways to reach a given number of conversions.

We simplify matters by using no spending limits. During the course of a day, the expected profit per impression can only increase as other agents hit their spending limits, and so it is desirable to receive all possible impressions in a day. Also, the cost paid per conversion tends to be higher at smaller positions – the slightly higher conversion rate at small positions is usually not enough to compensate for the higher CPC. It is therefore usually preferable to control conversions using the bid rather than the spending limit.[1]

For any given bid, we can evaluate each of the relevant ads and pick the one that gives the highest profit per conversion. As a result, the QA's primary task is to determine the bid that will result in the desired number of conversions. There is one difficulty remaining, however: because our prediction for each advertiser's bid is a point estimate, any bid between the $n$th and $n$+1st predicted bids will result in the same position, $n$+1, and the function mapping bids to conversions will be a step function. In reality, there is uncertainty about the bids of other advertisers, and we would expect this function to be continuous and monotonically increasing. We create such a function by linearly interpolating between the expected results for each position. In particular, we assume that the number of conversions expected for

[1]During the 2009 TAC/AA competition, TacTex used high spending limits as a precaution, but our experiments have shown that this was unnecessary, and so we omit them from the agent described here.

the $nth$ position will result from bidding the average of the $n$-1st and $n$th bid. For $n = 1$, we use a bid 10% above the predicted highest bid, and for $n = 8$, we use a bid 10% below the predicted lowest bid. We generate functions for cost and revenue in the same way.

The complete procedure followed by the QA for each query type $q$ is therefore as follows. First, we find the eight bids (along with corresponding optimal ads) that correspond to the eight possible positions, and determine the expected conversions, cost, and revenue for each. Next, we use linear interpolation to create functions mapping bids to conversions, cost, and revenue. Finally, for a conversion target $c$, we can find the bid resulting in the target from the conversions function and determine the resulting cost ($cost_q(c)$) and revenue ($revenue_q(c)$) from the corresponding functions. The ad to use is the ad corresponding to the closest of the eight bids.

## 8.2 Single-day Optimizer

Using this information about each query type, the SDO can now determine the optimal number of conversions to target for each query type given a total daily conversion target $c$ and the initial capacity used $u$. The initial capacity used (the sum from the past four days) is important because it, along with the total conversion target, determines the distribution constraint, which can in turn have a large impact on the profit from each conversion and the optimal solution.

Computing the precise impact of the distribution constraint is difficult because it decreases after each conversion, meaning that we would need to know when a conversion occurred to compute its profit. We solve this problem by making the simplifying assumption that the day's average distribution constraint applies to each conversion. We denote this value $\bar{d}(u, c)$ because it can be computed from the initial capacity used and the total conversion target; in fact, we precompute all possible $\bar{d}(u, c)$ values before the game begins. The goal of the SDO is thus to find values of $c_q$ maximizing $\sum_q [\bar{d}(u, c) revenue_q(c_q) - cost_q(c_q)]$, where the $c_q$ values correspond to the query types and sum to $c$. It is important to note that although we are targeting $c$ conversions, we would actually only expect $\bar{d}(u, c)c$ conversions. In general we reason in terms of conversions *before* adjusting for the distribution constraint, and so for clarity we will use the term *adjusted* conversions when referring to the actual number of resulting conversions. Note that $u$ is expressed in terms of adjusted conversions, while $c$ is not.

We are now left with a fairly straightforward optimization problem: allocating the total conversion target among the queries so as to maximize profit. The SDO solves this problem using a nearly-optimal greedy solution that repeatedly adds conversions from the most profitable query type. For each query type, we determine the number of additional conversions (bounded above by the number of conversions remaining before we hit our target) that maximizes the average profit per additional conversion. (Note that this number may be more than one, as the marginal profit per conversion is not necessarily monotonically decreasing.) We then add the conversions from the query type with the highest average profit. This greedy approach is not guaranteed to be optimal, but tests show that the resulting expected profit differs from the results of an optimal (but much slower) dynamic programming approach by less than 0.1% on average.

## 8.3 Multi-day Optimizer

The SDO determines bids for any given conversion target and amount of capacity already used. Determining the bids to submit for the next day therefore requires only that we choose the conversion target. Because the bids submitted today affect not only tomorrow's profit but also the capacity remaining on future days, we cannot simply choose the conversion target myopically. In order to maximize expected profit over the remainder of the game, we must consider the actions to be taken over all remaining days.

The MDO operates by finding the optimal set of conversion targets for the remainder of the game. The expected profit from any set of conversion targets can be determined by successively applying the SDO to each remaining game day. The goal of the MDO on day $d$ is is therefore to find the conversion targets $c_t$ maximizing $\sum_{t=d+1}^{59} SDO_t(c_t, u_t)$, where $SDO_t$ returns the expected profit from applying the SDO on day $t$, and $u_t$ represents the total adjusted conversions over four days preceding $t$ (which can be computed from the $c_t$ values). Note that planning for the entire game requires calling the QA (and thus predicting the bids of other agents) for all remaining game days, not only the next day. Currently, TacTex simply assumes that the bids predicted for the next day persist for the rest of the game, but improving these predictions is an important focus of future work.

The MDO uses a form of hill climbing search to solve this optimization problem. We begin by setting each $c_t$ value to be one-fifth of TacTex's capacity. Then for all $t$, we consider increasing or decreasing $c_t$ by one and compute the expected profit in each case. We then choose the most profitable deviation over all $t$. This process repeats until no deviation is profitable. Again, the performance of this approach is comparable to that of an optimal but slow dynamic programming approach. In fact, the greedy approach finds slightly better solutions due to the need to use a somewhat coarse degree of granularity (increments of five conversions) when implementing the dynamic programming approach because of memory limitations.

Once the optimal set of conversion targets is found, the MDO takes tomorrow's conversion target and submits the bids determined by the SDO. Essentially, we plan for the rest of the game and take the first step of this plan. On the next day, we repeat this process using updated information.

There is one remaining special case. It will often be the case that we are not interested in bidding on a particular query. When this happens, TacTex submits a probe bid designed to provide information about the bids of other advertisers. The bid chosen is one that we expect to be the $n$th ranked bid, where $n$ is the rank between 2 and 6 that we have hit least recently. We set a spending limit equal to the bid so that we will likely only receive a single click.

# 9. AGENT PERFORMANCE

Having completed the description of TacTex, we now analyze its performance. First, we present competition results. We then give the results of a number of controlled experiments designed to measure the importance of the different components of TacTex.

## 9.1 2009 Competition

The first annual TAC/AA competition was held over two days at IJCAI 2009. All 15 qualifying agents participated in a round-robin style semifinal round, and then the top eight agents advanced to a final round where each agent participated in all 80 games.

| *Agent* | *Score* |
|---|---|
| TacTex | 79.9 |
| AstonTAC | 76.2 |
| Schlemazl | 75.4 |
| QuakTAC | 74.5 |
| munsey | 71.8 |
| EPFLAgent | 71.7 |
| MetroClick | 70.6 |
| UMTac09 | 66.9 |

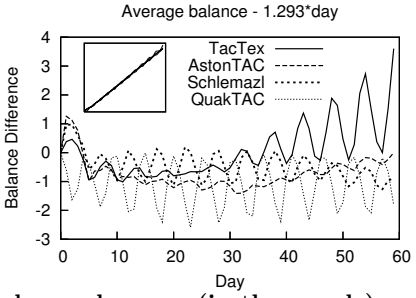


**Figure 2: 2009 final round scores (in thousands)**

TacTex finished first in both rounds. The scores (i.e., average ending balances) from the final round are shown in the table in Figure 2. A Wilcoxon two-sample test shows that the difference in score between TacTex and each other agent is statistically significant ($p < .05$ in each case). The plot in Figure 2 shows the average daily balance of the top four agents. Here we have subtracted 1293 per day (the average daily profit of the second place agent) from each balance to improve visibility – otherwise the plot would appear as nearly diagonal lines, as shown in the inset. Two things are apparent from this plot. First, scores tend to oscillate rather than increase smoothly. This oscillation is a consequence of the 5-day capacity window – if an agent has a large number of conversions on one day (frequently the very first day), it will have reduced capacity for four days, and then the heavy day will slide out of the window. There is a similar end game effect, as there is no benefit to saving capacity beyond the last day. TacTex is especially effective at ending on an upswing due to the Multi-day Optimizer. Second, TacTex does not pull away from the other agents until the second half of the game. One possible explanation is that bidding behaviors converge to some degree over the course of a game, and so TacTex's ability to make accurate predictions improves.

Although it is difficult to identify a main reason for TacTex's success, analysis of the game results from the finals does provide a few clues. Compared to other agents, TacTex typically had fewer conversions but had a higher profit per conversion (at least 6% higher than any other agent). AstonTAC and Schlemazl focused on selling those products that matched their manufacturer specialty, which gives a revenue bonus. As a result, their average revenue per conversion was slightly higher that TacTex's, but their clicks tended to come from higher positions and thus their CPCs were much higher than TacTex's. The reverse was true for all other agents: they had lower CPCs, but much lower average revenue per conversion. It appears that TacTex struck the right balance between targeting high revenue conversions and taking advantage of other sales opportunities.

## 9.2 Experiments

Although our competition victory suggests that the overall design of TacTex is sound, it gives us little information about the relative importance of the individual components. For instance, we cannot look at the results and determine how much the User Model contributed to the overall performance. Fortunately, after the competition, most teams submitted agent binaries to the TAC Agent Repository[2]. Using these binaries, we can experiment by changing certain portions of TacTex and observing the effect on performance. We now report on a number of these experiments.

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

| # | *Description* | *Diff.* | *p* |
|---|---|---|---|
| **Advertiser Model** | | | |
| 1 | bid estimator 1 | -284 | .150 |
| 2 | bid estimator 2 | -306 | .113 |
| 3 | bids * 0.9 | -536 | .077 |
| 4 | bids * 1.1 | -678 | .010 |
| 5 | bids * r(0.9,1.1) | -501 | .046 |
| 6 | all impressions | -3438 | .000 |
| 7 | impressions * .9 | -772 | .008 |
| **Other** | | | |
| 8 | no User Model | -896 | .001 |
| 9 | no Par. Model | -254 | .133 |
| 10 | no probing | -812 | .004 |

| # | *Description* | *Diff.* | *p* |
|---|---|---|---|
| **Optimization** | | | |
| 11 | maintain cap. | -4257 | .000 |
| 12 | constant conv. | -7362 | .000 |
| 13 | conv. * 0.9 | -287 | .174 |
| 14 | conv. * 1.1 | +62 | .596 |
| 15 | conv. * 1.2 | -1024 | .000 |
| 16 | bids * 0.9 | -453 | .039 |
| 17 | bids * 1.1 | -704 | .000 |
| 18 | bids * r(0.9,1.1) | -425 | .038 |
| 19 | all generic ads | -2351 | .000 |
| 20 | all targeted ads | -87 | .992 |

**Table 2: Experimental results**

We measure the impact of a change to TacTex by running two sets of 50 games: one with the original TacTex, and one with a modified version. The other seven agents are the best available agents from the repository. To improve our ability to evaluate the statistical significance of the results, we have modified the game server so that the main random factors (game parameters, advertiser capacities and specialties, and the random draws primarily responsible for determining user population transitions) are identical between corresponding pairs of games. This allows us to use the Wilcoxon matched-pairs signed-ranks test instead of an unpaired test. Table 2 shows the results of our experiments. For each experiment, we give a reference number, a brief description, the difference in score (a '-' means the score of the modified agent was lower), and the $p$-value indicating the significance. Experiments are divided into groups corresponding to different parts of TacTex.

The first group of experiments concerns the Advertiser Model. In experiments 1 and 2, we use only one of the bid estimators instead of averaging the two. Although both estimators appear to perform reasonably well on their own, we do see a small increase in score from combining them. In experiments 3-5, we multiply the Advertiser Model's predicted bids for each advertiser by 0.9, 1.1, or a random number chosen uniformly from the range [0.9, 1.1], respectively. Again, we see a small decrease in score from each change. We know from comparison with the true results that our predictions are not always highly accurate, but these experiments show that our predictions are helpful enough that small changes can impact performance negatively. It certainly does not appear to be the case that our predictions are consistently too high or too low. Finally, we look at the Advertiser Model's predictions of the number of impressions advertisers will receive. In experiment 6, we assume that advertisers use no spending limits (i.e., they receive all possible impressions). In experiment 7, when predicting that an advertiser will use a spending limit, we multiply the predicted impressions by 0.9. Both changes result in a large drop in score, showing that impression predictions are very important.

The next group of experiments do not fit a specific category. In experiment 8, we replace the predictions of the User Model with the average user populations expected and see a fairly large drop in score. Similarly, in experiment 9 we replace the estimates from the Parameter Model with the expected values and see a smaller score reduction. In experiment 10, we replace all probe bids with zero bids, and again we see that this is detrimental to performance. The information lost from these changes appears to be important to TacTex, and the last experiment shows that a query type that does not attract attention from TacTex on one day may

still become a source of profit in the future.

The last group of experiments concerns optimization. Experiments 11 and 12 show the importance of planning over multiple days. In experiment 11, we choose the next day's conversion target so that the total conversions over the last five days will be 1.4 times our capacity (the average amount used by the unaltered TacTex). Then in experiment 12, we simply set each day's conversion target to be 1/5 of 1.4 times the capacity. Both changes cause an extremely large drop in score despite the fact that TacTex has roughly the same number of conversions. In experiments 13-15, we multiply the next day's conversion target by 0.9, 1.1, and 1.2, respectively. Experiments 13 and 15 result in score reductions as expected; however, multiplying the conversion target by 1.1 actually results in an increase in score, although not a significant one. This result would make sense if TacTex were consistently receiving fewer conversions than expected, but that is not the case. Another possibility is that it may be less harmful to sell too much than too little, and because TacTex will rarely hit its conversion target exactly, it would be safer to target a few extra conversions. In any case, experiments 11-15 suggest that the Multi-day Optimizer is highly effective in choosing conversion targets. Experiments 16-18 concern the bids generated by the Query Analyzer at the end of the optimization process. Again, we multiply these bids by 0.9, 1.1, or a random number chosen uniformly from the range [0.9, 1.1], respectively. In each case there is a moderately large score reduction, and so the Query Analyzer appears to be successful at providing useful information about bid selection. Finally, experiments 19 and 20 concern ad selection. In experiment 19, we use only generic ads, and in experiment 20 we use only targeted ads, where the manufacturer and component match the query when present or TacTex's specialty when null. The results clearly show that ad selection can impact score, but it does not appear that choosing optimal ads (which are usually targeted) is significantly better than simply always using targeted ads.

The main message from these experiments is that nearly all components of TacTex make at least a small contribution to the overall performance of the agent. Some parts, such as the Multi-day Optimizer and impression prediction, appear to be especially important. In addition, from extra experiments not described here we have observed that changing multiple agent components tends to compound the impact on performance. For example, while using only the first bid estimator or multiplying our conversion targets by 0.9 both reduce the score by under 300, doing both at the same time reduces the score by 1238. The true value of some components may therefore be even greater than the experiments shown here suggest.

## 10. RELATED WORK

While keyword auctions have received considerable attention in recent years, there have been few published reports of complete agents that solve bidding problems of the scope faced in TAC/AA. One exception is an agent designed by Kitts and LeBlanc [3] to bid on a family of keywords in live Overture auctions. The agent bears a strong resemblance to TacTex at a high level and performs many of the same tasks, such as predicting the position and resulting profit for a given bid and optimizing bids over a time horizon.

Within the TAC/AA domain, there is currently limited information about the approaches taken by the other agents, but we are aware that they include decision-theoretic optimization similar to TacTex, rule-based systems, and genetic algorithms. Optimization-based approaches appeared to fare the best in general, and we believe that what sets TacTex apart in this area is its ability to estimate the full game state. This observation is consistent with past TAC competitions in the areas of travel arrangement [7] and supply chain management [1]. Successful agents tend to have similar optimization routines at their core, and the top agents are those that are best able to model their environment, with increasingly complex modeling approaches developed as the competitions mature.

## 11. CONCLUSION AND FUTURE WORK

In this paper we described TacTex, an agent for bidding in keyword auctions that is able to effectively estimate the full game state (including the actions of other advertisers) and optimize its actions. Competition results showed TacTex to be significantly more profitable than other competing agents, and our controlled experiments show that all components of TacTex played an important role in its performance.

Despite this success, there is still room for improvement in parts of TacTex. Our primary research agenda at this time is to apply machine learning to improve the Advertiser Model. By using the game data that is now available, we hope to be able to develop specific models of each advertiser, which will improve our bid estimators and allow us to better predict the actions that will be taken by other advertisers.

## 12. ACKNOWLEDGEMENTS

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).

## 13. REFERENCES

[1] J. Eriksson, N. Finne, and S. Janson. Evolution of a supply chain management game for the Trading Agent Competition. *AI Communications*, 19:1–12, 2006.

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

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

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

[5] 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.

[6] 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.

[7] M. P. Wellman, A. Greenwald, and P. Stone. *Autonomous Bidding Agents: Strategies and Lessons from the Trading Agent Competition*. MIT Press, 2007.