SCISPACE
formerly Typeset

Open access • Book Chapter • DOI:10.1007/3-540-36378-5_9

# ATTac-2001: A Learning, Autonomous Bidding Agent — Source link

Peter Stone, Robert E. Schapire, János A. Csirik, Michael L. Littman ...+1 more authors

**Institutions:** University of Texas at Austin, AT&T Labs, Rutgers University, Toyota

**Published on:** 16 Jul 2002 - Lecture Notes in Computer Science (Springer, Berlin, Heidelberg)

**Topics:** Bidding and Common value auction

Related papers:

- Algorithm Design for Agents which Participate in Multiple Simultaneous Auctions
- Autonomous Agents for Participating in Mulitple On-line Auctions
- Designing the market game for a trading agent competition
- Decision-theoretic bidding based on learned density models in simultaneous, interacting auctions
- A principled study of the design tradeoffs for autonomous trading agents

Share this paper:

View more about this paper here: https://typeset.io/papers/attac-2001-a-learning-autonomous-bidding-agent-ti9d2v227w

# ATTac-2001: A Learning, Autonomous Bidding Agent

Peter Stone*, Robert E. Schapire†, János A. Csirik†,
Michael L. Littman, David McAllester

*Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712-1188

†AT&T Labs — Research
180 Park Avenue
Florham Park, NJ 07932

pstone@cs.cmu.edu, {schapire, janos}@research.att.com,
mlittman@cs.duke.edu, mcallester@autoreason.com

**Abstract.** Auctions are becoming an increasingly popular method for transacting business, especially over the Internet. This paper presents a general approach to building autonomous bidding agents to bid in multiple simultaneous auctions for interacting goods. The core of our approach is learning a model of the empirical price dynamics based on past data and using the model to analytically calculate, to the greatest extent possible, optimal bids. This approach is fully implemented as ATTac-2001, a top-scoring agent in the second Trading Agent Competition (TAC-01). ATTac-2001 uses boosting techniques to learn conditional distributions of auction clearing prices. We present experiments demonstrating the effectiveness of this predictor relative to several reasonable alternatives.

## 1 Introduction

Auctions are becoming an increasingly popular method for transacting business, especially over the Internet. In an auction for a single good, it is straightforward to create automated bidding strategies—an agent could keep bidding until reaching a target reserve price, or it could monitor the auction and place a winning bid just before the closing time (known as *sniping*).

When bidding for multiple interacting goods in simultaneous auctions, however, agents must be able to reason about uncertainty and make complex value assessments. For example, an agent bidding for a camera and a flash may end up buying the flash and then not being able to find an affordable camera. Alternatively, if bidding for the same good in several auctions, it may purchase two flashes when only one was needed.

This paper presents ATTac-2001, a top-scoring agent[1] in the second Trading Agent Competition (TAC-01), which was held in Tampa Bay, FL on October 14, 2001. We present details of the agent as well as underlying principles that we believe have applications in a wide variety of bidding situations.

ATTac-2001 bids simultaneously for multiple interacting goods. We start with the observation that the key challenge in auctions is the prediction of eventual prices of goods: with complete knowledge of eventual prices, there are direct methods for determining the optimal bids to place. Our guiding principle is to have the agent model its *uncertainty* in eventual prices and, to the greatest extent possible, analytically calculate optimal bids. ATTac-2001 uses a predictive, data-driven approach to bidding based on

[1] Top-scoring by one metric, and second place by another.

expected marginal values of all available goods. A price-predictor based on boosting techniques is at the heart of the algorithm.

In this paper, we present empirical results demonstrating the robustness and effectiveness of ATTac-2001's adaptive strategy. We also report on ATTac-2001's performance at TAC-01 and reflect on some of the key issues raised during the competition. The remainder of the paper is organized as follows. In Section 2, we present our general approach to bidding for multiple interacting goods in simultaneous auctions. In Section 3, we summarize the substrate domain for our work, TAC. In Section 4, we give the details of ATTac-2001, with special emphasis on the machine-learning-based price-predictor. In Section 5, we present empirical results including a summary of ATTac-2001's performance in TAC-01; controlled experiments isolating the successful aspects of ATTac-2001; and controlled experiments illustrating some of the lessons learned during the competition. Section 6 presents related work and concludes.

## 2 General Approach

In a wide variety of decision-theoretic settings, it is useful to be able to evaluate hypothetical situations. In computer chess, for example, a static board evaluator is used to heuristically measure which player is ahead and by how much in a given board situation. ATTac-2001 uses a situation evaluator, analogous to the static board evaluator, which estimates the agent's expected profit in a hypothetical future situation. This "profit-predictor" has a wide variety of uses in the agent. For example, to determine the value of an item, the agent compares the predicted profit assuming the item is already owned to the predicted profit assuming that the item is not available.

Given prices for goods, one can compute a set of purchases and an allocation that maximizes profit.[2] Similarly, if closing prices are known, they can be treated as fixed and optimal bids can be computed (bid high for anything you want to buy). So one natural profit predictor is simply to calculate the profit of optimal purchases under fixed predicted prices.

A more sophisticated approach to profit-prediction, used in our agent, is to construct a model of the probability distribution over possible future prices; stochastically sample possible prices; and compute a profit prediction, as above, for each sampled price. This more sophisticated form of profit-prediction is important for modeling uncertainty and the value of gaining information, i.e., reducing the price uncertainty. In short, the guiding principle in ATTac-2001 is to make decision-theoretically optimal decisions given profit-predictions for hypothetical future situations[3].

Our approach to profit-prediction is based upon four simplifying assumptions:

[2] The problem appears computationally difficult in general, but was solved effectively in practice in TAC-00 [7, 19].

[3] An alternative approach would be to abstractly calculate the Bayes-Nash equilibrium [8] for the game and play the optimal strategy. We dismissed this approach because of its intractability in situations as complex as TAC. Furthermore, even if we were able to approximate the equilibrium strategy, it is reasonable to assume that our opponents would not play their optimal strategy either. Thus we could gain additional advantage by tuning our approach to our opponents' *actual* behavior as observed in the earlier rounds.

1. Closing prices are somewhat, but only somewhat, predictable.
2. Our own bids do not have an appreciable effect on the economy.
3. The agent is free to change existing bids in auctions that have not yet closed.
4. Future decisions are made in the presence of complete price information.

While these assumptions are not all true in general, they can be reasonable enough approximations to be the basis for an effective strategy.

By Assumption 2, the price-predictor can generate predicted prices prior to considering one's bids. Thus, we can sample from these distributions to produce complete sets of closing prices of all goods.

For each good under consideration, we assume that it is the next one to close. If a different auction closes first, we can then revise our bids later (Assumption 3). Thus, we would like to bid exactly the good's *expected marginal utility* to us. That is, we bid the difference between the expected utilities attainable with and without the good. To compute these expectations, we simply average the utilities of having and not having the good under different price samples. This strategy rests on Assumption 4 in that we assume that bidding the good's current expected marginal utility cannot adversely affect our future actions.

Note that as time proceeds, the price distributions change in response to the observed price trajectories, thus causing the agent to continually revise its bids. By using this strategy, provided that the price is right, the agent automatically buys contingency goods to guard against the possibility of the most desired goods becoming too expensive: for price samples in which the most desired goods are expensive, the contingency goods will have marginal value, thus leading to a positive *expected* marginal value.

A more detailed description of the agent is given in Section 4.

## 3 TAC

We instantiated our approach as an entry in the second Trading Agent Competition (TAC), as described in this section. Building on the success of TAC-00 held in July 2000 [21], TAC-01 included 19 agents from 9 countries. A key feature of TAC is that it required *autonomous bidding agents* to buy and sell *multiple interacting goods* in auctions of different types. It is designed as a benchmark problem in the complex and rapidly advancing domain of e-marketplaces, motivating researchers to apply unique approaches to a common task. By providing a clear-cut objective function, TAC also allows the competitors to focus their attention on the computational and game theoretic aspects of the problem and leave aside the modeling and model validation issues that invariably loom large in real applications of automated agents to auctions (see [15]). Another feature of TAC is that it provides an academic forum for open comparison of agent bidding strategies in a complex scenario, as opposed to other complex scenarios, such as trading in real stock markets, in which practitioners are (understandably) reluctant to share their technologies.

A TAC game instance pits eight autonomous bidding agents against one another. Each TAC agent is a simulated travel agent with eight clients, each of whom would like to travel from TACtown to Tampa and back again during a 5-day period. Each client is characterized by a random set of preferences for the possible arrival and departure dates;

hotel rooms; and entertainment tickets. To satisfy a client, an agent must construct a travel package for that client by purchasing airline tickets to and from TACtown and securing hotel reservations; it is possible to obtain additional bonuses by providing entertainment tickets as well. A TAC agent's score in a game instance is the difference between the sum of its clients' utilities for the packages they receive and the agent's total expenditure. We provide selected details about the game next; for full details on the design and mechanisms of the TAC server and TAC game, see `http://tac.eecs.umich.edu`.

TAC agents buy flights, hotel rooms and entertainment tickets through auctions, run from the TAC server at the University of Michigan. Each game instance lasts 12 minutes and includes a total of 28 auctions of 3 different types.

**Flights (8 auctions):** There is a separate auction for each type of airline ticket: to Tampa (*inflights*) on days 1–4 and from Tampa (*outflights*) on days 2–5. There is an unlimited supply of airline tickets, and their ask price changes randomly every 30 seconds or so, with an increasing bias upwards. When the server receives a bid at or above the ask price, the transaction is cleared immediately at the ask price and no resale is allowed.

**Hotel Rooms (8):** There are two different types of hotel rooms—the Tampa Towers (TT) and the Shoreline Shanties (SS)—each of which has 16 rooms available on days 1–4. The rooms are sold in a 16th-price *ascending* (English) auction, meaning that for each of the 8 types of hotel rooms, the 16 highest bidders get the rooms at the 16th highest price. For example, if there are 15 bids for TT on day 2 at $300, 2 bids at $150, and any number of lower bids, the rooms are sold for $150 to the 15 high bidders plus one of the $150 bidders (earliest received bid). The ask price is the current 16th-highest bid and transactions clear only when the auction closes. No bid withdrawal or resale is allowed, though the price of bids may be lowered provided the agent does not reduce the number of rooms it would win were the auction to close. One *randomly chosen* hotel auction closes at minutes 4–11 of the 12-minute game.

**Entertainment Tickets (12):** Alligator wrestling, amusement park, and museum tickets are each sold for days 1–4 in continuous double auctions. Here, agents can *buy and sell* tickets, with transactions clearing immediately when one agent places a buy bid at a price at least as high as another agent's sell price. Unlike the other auction types in which the goods are sold from a centralized stock, each agent starts with a (skewed) random endowment of entertainment tickets.

In addition to unpredictable market prices, other sources of variability from game instance to game instance are the client profiles assigned to the agents and the random initial allotment of entertainment tickets. Each TAC agent has eight clients with randomly assigned travel preferences. Clients have parameters for ideal arrival day, *IAD* ; ideal departure day, *IDD* ; hotel premium, *HP* ; and entertainment values, *EV* for each type of entertainment ticket.

The utility obtained by a client is determined by the travel package that it is given in combination with its preferences. To obtain a non-zero utility, the client must be assigned a *feasible* travel package consisting of an inflight on some arrival day *AD*, an outflight on a departure day *DD*, and hotel rooms of the *same type* (TT or SS) for the

days in between. At most one entertainment ticket of each type can be assigned, and no more than one on each day. Given a feasible package, the client's utility is defined as

$$1000 - \textit{travelPenalty} + \textit{hotelBonus} + \textit{funBonus}$$

where

- *travelPenalty* = $100(|AD - IAD| + |DD - IDD|)$
- *hotelBonus* = *HP* if the client is in the TT, 0 otherwise.
- *funBonus* = sum of *EV*s for assigned entertainment tickets.

A TAC agent's *score* is the sum of its clients' utilities in the optimal allocation of its goods (computed by the TAC server) minus its expenditures.

TAC-01 was organized as a series of four competition phases, culminating with the semifinals and finals on 14 October 2001 at the EC-01 conference in Tampa, Florida. First, the qualifying round, consisting of about 270 games per agent, served to select the 16 agents that would participate in the semifinals. Second, the seeding round, consisting of about 315 games per agent, was used to divide these agents into two groups of eight. After the semifinals, on the morning of the the 14th consisting of 11 games in each group, four teams from each group were selected to compete in the finals during that same afternoon. The finals are summarized in Section 5.

## 4 ATTac-2001

This section presents the details of ATTac-2001's algorithm. We begin with a brief description of the goods allocator, which is used as a subroutine throughout the algorithm. We then present the algorithm in a top-down fashion.

### 4.1 Starting Point

A core subproblem for TAC agents is the allocation problem: finding the most profitable allocation of goods to clients, $G^*$, given a set of owned goods and prices for all other goods. We denote the value of $G^*$ (i.e. the score one would attain with $G^*$) as $v(G^*)$. The general allocation problem is NP-complete, as it is equivalent to the set-packing problem [5]. However it can be solved tractably in TAC via integer linear programming [19].

The solution to the integer linear program is a value-maximizing allocation of owned resources to clients along with a list of resources that need to be purchased. Using the linear programming package "LPsolve", ATTac-2001 is usually able to find the globally optimal solution in under less than 0.01 seconds on a 650 MHz Pentium II. However, since integer linear programming is an NP-complete problem, some inputs can lead to a great deal of search over the integrality constraints, and therefore significantly longer solution times. When only $v(G^*)$is needed (as opposed to $G^*$itself), the upper bound produced by LPsolve prior to the search over the integrality constraints, known as the LP relaxation, can be used as an estimate. The LP relaxation can always be generated very quickly[4].

[4] The LP is described in full detail in [19].

Note that this is not by any means the only possible formulation of the allocation. [7] studied a fast, heuristic variant and found that it performed extremely well on a collection of large, random allocation problems. [19] used a randomized greedy strategy as a fallback for the cases in which the linear program took too long to solve.

Table 1 shows a high-level overview of ATTac-2001. The italicized portions are described in the remainder of this section.

| |
|---|
| **When the first flight quotes are posted:** |
| • Compute $G^*$ with current holdings and *expected prices* |
| • Buy the flights in $G^*$ for which *expected cost of postponing commitment* exceeds the *expected benefit of postponing commitment* |
| **Starting 1 minute before each hotel close:** |
| • Compute $G^*$ with current holdings and *expected prices* |
| • Buy the flights in $G^*$ for which *expected cost of postponing commitment* exceeds *expected benefit of postponing commitment* (30 seconds) |
| • Bid *hotel room expected marginal values* given holdings, new flights, and *expected hotel purchases* (30 seconds) |
| **Last minute:** Buy remaining flights as needed by $G^*$ |
| **In parallel (continuously):** Buy/sell entertainment tickets based on their *expected values* |

**Table 1**: ATTac-2001's high-level algorithm. The italicized portions are described in the remainder of this section.

### 4.2 Hotel Price Prediction

As discussed earlier, a central part of our strategy depends on the ability to predict hotel prices at various points in the game. To do this as accurately as possible, we used machine-learning techniques that would examine the hotel prices actually paid in previous games to predict prices in future games.

There is bound to be considerable uncertainty regarding hotel prices since these depend on many unknown factors, such as the time at which the hotel room will close, who the other agents are, what kind of clients have been assigned to each agent, etc. Thus, *exactly* predicting the price of a hotel room is hopeless. Instead, we regard the closing price as a random variable that we need to estimate conditional on our current state of knowledge (i.e., number of minutes remaining in the game, ask price of each hotel, flight prices, etc.). We might then attempt to predict this variable's conditional expected value. However, our strategy requires that we not only predict expected value, but that we also be able to estimate the *entire* conditional distribution so that we can *sample* hotel prices.

To set this up as a learning problem, we gathered a set of training examples from previously played games. We defined a set of features for describing each example that together are meant to comprise a snap-shot of all the relevant information available at the time each prediction is made. We used the following basic features: number of minutes remaining in the game; the current ask price for rooms that have not closed

or the actual selling price for rooms that have; the closing time of each hotel room;[5] all flight prices. To this basic list, we added a number of redundant variations that we thought might help the learning algorithm. During the semifinals and finals when we knew the identities of all our competitors, we also added features indicating the number and identities of each of the players.

We trained specialized predictors for predicting the price of each type of hotel room. We also created separate predictors for predicting in the first minute after flight prices had been quoted but prior to receiving any hotel price information. We trained our predictors to predict not the actual closing price of each room per se, but rather how much the price would increase, a quantity that we thought might be more predictable.

From each of the previously played games, we were able to extract many examples, in particular, for each minute of the game and for each room that had not yet closed, the values of all of the features described above at that moment in the game, plus the actual closing price of the room.

We solved this learning problem by first reducing to a multiclass, multi-label classification problem (or alternatively a multiple logistic regression problem), and by then applying boosting techniques developed by Schapire and Singer [16] combined with a modification of boosting algorithms for logistic regression proposed by Collins, Schapire and Singer [2]. The resulting technique turns out to be a new supervised learning algorithm for solving conditional density estimation problems. Full details of this machine learning algorithm are presented in [17].

### 4.3 Cost of Additional Rooms

Our hotel price predictor described above assumes that ATTac-2001's bids do not affect the ultimate closing price (Assumption 2). This assumption holds in a large economy. However in TAC, each hotel auction involved $8$ agents competing for $16$ hotel rooms. Therefore, the actions of each agent had an appreciable effect on the clearing price: the more hotel rooms an agent attempted to purchase, the higher the clearing price would be, all other things being equal. This effect needed to be taken into account when solving the basic allocation problem.

The simplified model used by ATTac-2001 assumed that the $n$th highest bid in a hotel auction was roughly $kc^{-n}$ (over the appropriate range of $n$) for a universal constant $c \geq 1$ that was the same for all auctions, and another constant $k$ which was different for each room auction. Thus, if the predictor gave a price of $p$, ATTac-2001 only used this for purchasing two hotel rooms (the "fair" share of a single agent of the $16$ rooms), and adjusted prices for other quantities of rooms by using $c$. For example, if ATTac-2001 wanted to obtain $4$ rooms, it would consider the predicted price to be $p = kc^{-16}$, the sixteenth highest price, and would take the clearing price to be $kc^{-14} =$

[5] Although there is no problem extracting closing times of all rooms during training, during actual play, we do not know closing times of rooms that have not yet closed. However, we do know the exact distribution of closing times of all of the rooms that have not yet closed. Therefore, to sample a vector of hotel prices, we can first sample according to this distribution over closing times, and then use our predictor to sample hotel prices using these sampled closing times.

$pc^2$, since with the extra two units of demand from ATTac-2001, the clearing price will be the the former fourteenth highest price (the total cost for all four rooms is of course $4pc^2$).

The constant $c$ was calculated from the data of several hundred games during the seeding round. In each hotel auction, the ratio of the $14$th and $18$th highest bids (reflecting the most relevant range of $n$) was taken as an estimate of $c^4$, and the (geometric) mean of the resulting estimates was taken to obtain $c = 1.35$.

The LP allocator takes these price estimates into account when computing $G^*$, thus tending to spread out ATTac-2001's demand over the different hotel auctions.

### 4.4 Hotel Expected Marginal Values

Using the hotel price prediction module modified as described above, ATTac-2001 is equipped to determine its bids for hotel rooms.

Every minute, for each hotel auction that is still open, ATTac-2001 assumes that that auction will close next and computes the marginal value of that hotel room given the predicted closing prices of the other rooms. If the auction does not close next, then it assumes that it will have a chance to revise its bids. Since these predicted prices are represented as distributions of possible futures, ATTac-2001 samples from these distributions and averages the marginal values to obtain an expected marginal value. Using the full minute between closing times for computation (or 30 seconds if there are still flights to consider too), ATTac-2001 divides the available time among the different open hotel auctions and generates as many price samples as possible for each hotel room. In the end, ATTac-2001 bids the expected marginal values for each of the rooms.

The algorithm is described precisely and with explanation in Table 2.

- For each hotel (in order of increasing expected price):
- Repeat until time bound
  1. Generate a random hotel closing order (only other open hotels)
  2. *Sample* closing prices from predicted hotel price distributions
  3. Given these closing prices, compute $V_0, V_1, \ldots V_n$
     - $V_i \equiv v(G^*)$if owning $i$ of the hotel
     - Estimate $v(G^*)$with LP relaxation
     - Assume that no additional hotel rooms of this type can be bought
     - For other hotels, assume outstanding bids above sampled price are already owned (i.e. they cannot be withdrawn).
     - Note that $V_0 \leq V_1 \leq \ldots \leq V_n$: the values are monotonically increasing since having more goods cannot be worse in terms of possible allocations.
- The value of the $i$th copy of the room is the mean of $V_i - V_{i-1}$ over all the samples.
- Note further that $V_1 - V_0 \geq V_2 - V_1 \ldots \geq V_n - V_{n-1}$: the values differences are monotonically decreasing since each additional room will be assigned to the client who can derive the most value from it.
- Bid for one room at the value of the $i$th copy of the room for all $i$ such that the value is at least as much as the current price. Due to the monotonicity noted in the step above, no matter what the closing price, the desired number of rooms at that price will be purchased.

**Table 2**: The algorithm for generating bids for hotel rooms.

One additional complication regarding hotel auctions is that, contrary to one of our assumptions, bids are not fully retractable: they can only be changed to $1 above the current ask price. In the case that there are current active bids for goods that ATTac-2001 no longer wants that are less than $1 above the current ask price, it may be advantageous to refrain from changing the bid in the hopes that the ask price will surpass them: that is, the current bid may have a higher expected value than the best possible new bid. To address this issue, ATTac-2001 samples from the learned price distribution to find the average expected values of the current and potential bids, and only enters a new bid in the case that the potential bid is better.

### 4.5 Expected Cost/Benefit of Postponing Commitment

ATTac-2001 makes flight bidding decisions based on a cost-benefit analysis: in particular, ATTac-2001 computes the incremental cost of postponing bidding for a particular flight versus the value of delaying commitment. ATTac-2001 took the cost of postponing commitment to be the average predicted cost of the flight over the next *flight-lookahead* whole minutes. It computed this cost based on a knowledge of the general form of the price adjustment function and the observed price points thus far in the current game. During TAC-01, ATTac-2001 started with the *flight-lookahead* parameter set to 3 (i.e., cost of postponing is the average of the predicted flight costs 1, 2, and 3 minutes in the future). However, this parameter was changed to 2 by the end of the finals in order to cause ATTac-2001 to delay its flight commitments further.

Fundamentally, the benefit of postponing commitments to flights is that additional information about the eventual hotel prices becomes known. Thus, the benefit of postponing commitment is computed by sampling possible future price vectors and determining, on average, how much better the agent could do if it bought a different flight instead of the one in question. If it is optimal to buy the flight in all future scenarios, then there is no value to delaying the commitment and the flight is purchased immediately. However, if there are many scenarios in which the flight is not the best one to get, the purchase is more likely to be delayed.

The algorithm for determining the benefit of postponing commitment is similar to that for determining the marginal value of hotel rooms. It is detailed in Table 3.

### 4.6 Entertainment Expected Values

The core of ATTac-2001's entertainment-ticket-bidding strategy is again a calculation of the expected marginal values of each ticket. For each ticket, ATTac-2001 computes the expected value of having one more and one fewer of the ticket. These calculations give bounds on the bid and ask prices it is willing to post. The actual bid and ask prices are a linear function of time remaining in the game: ATTac-2001 settles for a smaller and smaller profit from ticket transactions as the game goes on. Details of the functions of bid and ask price as a function of game time and ticket value remained unchanged from the previous year's agent [19].

- Assume we're considering buying $n$ flights of a given type
- Repeat until time bound
  1. Generate a random hotel closing order (open hotels)
  2. *Sample* closing prices from predicted price distributions (open hotels)
  3. Given these closing prices compute $V_0, V_1, \ldots V_n$
     - $V_i$ = $v(G^*)$if forced to buy $i$ of the flight
     - Estimate $v(G^*)$with LP relaxation
     - Assume more flights can be bought at the *current price*
     - Note that $V_0 \geq V_1 \geq \ldots \geq V_n$ since it is never worse to retain extra flexibility.
- The value of waiting to buy copy $i$ is the mean of $V_i - V_{i-1}$ over all the samples. If all price samples lead to the conclusion that the $i$th flight should be bought, then $V_i = V_{i-1}$ and there is no benefit to postponing commitment.

**Table 3**: The algorithm for generating value of postponing flight commitments.

# 5 Results

## 5.1 TAC-01 Competition

Of the 19 teams that entered the qualifying round, ATTac-2001 was one of eight agents to make it to the finals on the afternoon of October 14th, 2001. The finals consisted of 24 games among the same eight agents. Right from the beginning, it became clear that livingagents [4] was the team to beat in the finals. They jumped to an early lead in the first two games, and by eight games into the round, they were more than 135 points per game ahead of their closest competitor (SouthamptonTAC [9]). 16 games into the round, they were more than 250 points ahead of their two closest competitors (ATTac-2001 and whitebear).

From that point, ATTac-2001 which was continually retraining its price-predictors based on recent games, began making a comeback. By the time the last game was to be played, it was only an average of 22 points per game behind livingagents. It thus needed to beat livingagents by 514 points in the final game to overtake it, well within the margins observed in individual game instances. As the game completed, ATTac-2001's score of 3979 was one of the first scores to be posted by the server. The other agents' scores were reported one by one, until only the livingagents score was left. After agonizing seconds (at least for us), the TAC server posted a final game score of 4626, enough for livingagents to retain the lead.

After the competition, the TAC team at the University of Michigan conducted a regression analysis of the effects of the client profiles on agent scores. Using data from the seeding rounds, it was determined that agents did better when their clients had:

1. fewer total preferred travel days;
2. higher total entertainment values; and
3. a higher ratio of outer days (1 and 4) to inner (2 and 3) in preferred trip intervals.

Based on these significant measures, the games in the finals could be handicapped based on each agent's aggregate client profiles. Doing so indicated that livingagents' clients were much easier to satisfy than those of ATTac-2001, giving ATTac-2001 the highest handicapped score. The final scores, and the handicapped scores, are shown in Table 4. Complete results and affiliations are at `http://tac.eecs.umich.edu`.

| Agent | Mean | Handicapped score |
|---|---|---|
| ATTac-2001 | 3622 | 4154 |
| livingagents | 3670 | 4094 |
| whitebear | 3513 | 3931 |
| Urlaub01 | 3421 | 3909 |
| Retsina | 3352 | 3812 |
| CaiserSose | 3074 | 3766 |
| SouthamptonTAC | 3253* | 3679 |
| TacsMan | 2859 | 3338 |

**Table 4**: Scores during the finals. Each agent played 24 games. *SouthamptonTAC's score was adversely affected by a game in which it crashed, leading to a loss of over 3000 points. Discounting that game would have led to an average score of 3531.

### 5.2 Controlled Experiments

ATTac-2001's success in the competition demonstrates its effectiveness as a complete system. However, since the different agents differ along several dimensions, the competition results cannot isolate the successful approaches. In this section we report on controlled experiments designed to test the efficacy of ATTac-2001's machine-learning approach to price prediction.

**Varying the Predictor** In the first set of experiments, we attempted to determine how the quality of ATTac-2001's hotel price predictions affects its performance. To this end, we devised seven price prediction schemes, varying considerably in sophistication and inspired by approaches taken by other TAC competitors, and incorporated these schemes into our agent. We then played these seven agents against one another repeatedly, with regular retraining as described below.

Following are the seven hotel prediction schemes that we used, in decreasing order of sophistication:

- ATTac-2001$_s$: This is the "full-strength" agent based on boosting that was used during the tournament.
- ConditionalMean$_s$: This agent samples prices from the empirical distribution of prices from previously played games, conditioned on the closing time of the hotel room. In other words, it collects all historical hotel prices and breaks them down by the time at which the hotel closed (as well as room type, as usual). The price predictor then simply samples from the collection of prices corresponding to the given closing time.
- SimpleMean$_s$: This agent samples prices from the empirical distribution of prices from previously played games, without regard to the closing time of the hotel room (but still broken down by room type).
- ATTac-2001$_{ns}$ ConditionalMean$_{ns}$, SimpleMean$_{ns}$: These agents predict in the same way as their corresponding predictors above, but instead of returning a random sample from the estimated distribution of hotel prices, they deterministically return the expected value of the distribution.

– CurrentBid: This agent uses a very simple predictor that always predicts that the hotel room will close at its current price.

In every case, whenever the price predictor returns a price that is below the current price, we replace it with the current price (since prices cannot go down).

In our experiments, we added as an eighth agent EarlyBidder, inspired by the livingagents agent. EarlyBidder used SimpleMean$_{ns}$, determined an optimal set of purchases, and then, right after the first flight quotes, placed bids for these goods at sufficiently high prices to ensure that they would be purchased. It never revised these bids.

Each of these agents require training, i.e., data from previously played games. However, we are faced with a sort of "chicken and egg" problem: to run the agents, we need to first train the agents using data from games involving the agent, but to get this kind of data, we need to first run the agents. To get around this problem, we ran the agents in phases. In Phase I, consisting of 126 games, we used training data from the seeding, semifinals and finals rounds of TAC-01. In Phase II, lasting 157 games, we retrained the agents once every six hours using all of the data from the seeding, semifinals and finals rounds as well as all of the games played so far during the course of the experiment. Finally, in Phase III, lasting 622 games, we continued to retrain the agents once every six hours, but now using only data from games played during the course of the experiment, and not including data from the seeding, semifinals and finals rounds. Note that the although each game involves the same agents, most of the agents are continuously adapting their price predictors. Thus the economy does not necessarily stabilize.

| *Agent* | *Relative Score* | | |
|---|---|---|---|
| | *Phase I* | *Phase II* | *Phase III* |
| ATTac-2001$_{ns}$ | $105.2 \pm 49.5$ (2) | $131.6 \pm 47.7$ (2) | $166.2 \pm 20.8$ (1) |
| ATTac-2001$_{s}$ | $27.8 \pm 42.1$ (3) | $86.1 \pm 44.7$ (3) | $122.3 \pm 19.4$ (2) |
| EarlyBidder | $140.3 \pm 38.6$ (1) | $152.8 \pm 43.4$ (1) | $117.0 \pm 18.0$ (3) |
| SimpleMean$_{ns}$ | $-28.8 \pm 45.1$ (5) | $-53.9 \pm 40.1$ (5) | $-11.5 \pm 21.7$ (4) |
| SimpleMean$_{s}$ | $-72.0 \pm 47.5$ (7) | $-71.6 \pm 42.8$ (6) | $-44.1 \pm 18.2$ (5) |
| Cond'lMean$_{ns}$ | $8.6 \pm 41.2$ (4) | $3.5 \pm 37.5$ (4) | $-60.1 \pm 19.7$ (6) |
| Cond'lMean$_{s}$ | $-147.5 \pm 35.6$ (8) | $-91.4 \pm 41.9$ (7) | $-91.1 \pm 17.6$ (7) |
| CurrentBid | $-33.7 \pm 52.4$ (6) | $-157.1 \pm 54.8$ (8) | $-198.8 \pm 26.0$ (8) |

**Table 5**: The average relative scores for eight agents in the three phases of a controlled experiment in which the hotel prediction algorithm was varied. The relative score of an agent is its score minus the average score of all agents in that game. The agent's rank within each phase is shown in parentheses.

Table 5 shows how the agents performed in each of these phases. Much of what we observe in this table is consistent with our expectations. As expected, the more sophisticated boosting-based agents (ATTac-2001$_s$ and ATTac-2001$_{ns}$) clearly dominated the agents based on simpler prediction schemes. Moreover, with continued training, these agents improved markedly relative to EarlyBidder. We also see the performance of the simplest agent, CurrentBid, which does not employ any kind of training, significantly decline relative to the other data-driven agents.

On the other hand, there are some phenomena in this table that were very surprising to us. Most surprising was the failure of sampling to help. Our strategy relies heavily not only on estimating hotel prices, but also on taking samples from the distribution of hotel prices. Yet these results indicate that using expected hotel price, rather than price samples, consistently performs better. We speculate that this may be because an insufficient number of samples are being used (due to computational limitations) so that the numbers derived from these samples have too high a variance. Another possibility is that the method of using samples to estimate scores consistently overestimates the expected score because it assumes the agent can behave with perfect knowledge for each individual sample—a property of our approximation scheme. Finally, as our algorithm uses sampling at several different points (computing hotel expected values, deciding when to buy flights, pricing entertainment tickets, etc.), it is quite possible that sampling is beneficial for some decisions while detrimental for others. In ongoing research, we are conducting experiments to isolate these issues.

We were also surprised that ConditionalMean$_s$ and ConditionalMean$_{ns}$ eventually performed worse than the less sophisticated SimpleMean$_s$ and SimpleMean$_{ns}$. One possible explanation is that the simpler model happens to give predictions that are just as good as the more complicated model, perhaps because closing time is not terribly informative, or perhaps because the adjustment to price based on current price is more significant. The simpler model has the additional advantage that its statistics are based on all of the price data, regardless of closing time, whereas the conditional model makes each prediction based on only an eighth of the data (since there are eight possible closing times, each equally likely).

In addition to agent performance, it is possible to measure the inaccuracy of the eventual predictions, at least for the non-sampling agents[6]. For these agents, we measured the root mean squared error of the predictions made in Phase III. These were: 56.0 for ATTac-2001$_{ns}$, 66.6 for SimpleMean$_{ns}$, 69.8 for CurrentBid and 71.3 for ConditionalMean$_{ns}$. Thus, we see that the more accurate the predictions (according to this measure), the higher the score.

**ATTac-2001 vs. EarlyBidder** There is an interesting contrast between the two agents that finished at the top of the standings in TAC-01. livingagents uses a simple open-loop strategy, committing to a set of desired goods right at the beginning of the game, while ATTac-2001 uses a closed-loop, adaptive strategy.

The open-loop strategy relies on the other agents to stabilize the economy and create consistent final prices. In this sense, the open-loop strategy is a parasite strategy. Table 6 shows the results of running 27 games with 7 copies of the open-loop EarlyBidder. In the experiments, one copy of ATTac-2001$_s$ is included for comparison. The price predictors are all from Phase I in the preceding experiments. EarlyBidder's high bidding strategy backfires and it ends up overpaying significantly for its goods.

Compared to the open-loop strategy, ATTac-2001's strategy is relatively stable against itself. Its main drawback is that as it changes its decision about what goods it wants and

[6] It is not clear how to measure the accuracy of the sampling agents' predictors since they generate distributions, while only a single actual measurement (final price) is available.

| Agent | Score | Utility |
|---|---|---|
| ATTac-2001 | 2431 ± 464 | 8909 ± 264 |
| EarlyBidder(7) | -4880 ± 337 | 9870 ± 34 |

**Table 6**: The results of running ATTac-2001 against 7 copies of EarlyBidder over the course of 27 games.

as it may also buy goods to hedge against possible price changes, it can end up getting stuck paying for some goods that are ultimately useless to any of its clients.

Table 7 shows the results of 7 copies of ATTac-2001 playing against each other and one copy of the EarlyBidder[7]. Again, training is from the seeding round and finals of TAC-01: the agents don't adapt during the experiment. Included in this experiment are three variants of ATTac-2001, each with a different *flight-lookahead* parameter (from the section on "cost of postponing flight commitments"). There were three copies each of the agents with *flight-lookahead* set to 2 and 3 (ATTac-2001(2) and ATTac-2001(3) respectively), and 1 ATTac-2001 agent with *flight-lookahead* set to 4.

| Agent | Score | Utility |
|---|---|---|
| EarlyBidder | 2869 ± 69 | 10079 ± 55 |
| ATTac-2001(2) | 2614 ± 38 | 9671 ± 32 |
| ATTac-2001(3) | 2570 ± 39 | 9641 ± 32 |
| ATTac-2001(4) | 2494 ± 68 | 9613 ± 55 |

**Table 7**: The results of running the EarlyBidder against 7 copies of ATTac-2001 over the course of 197 games. The three different versions of ATTac-2001 had slightly different *flight-lookahead*s.

From the results in Table 7 it is clear that ATTac-2001 does better when committing to its flight purchases later in the game (ATTac-2001(2) as opposed to ATTac-2001(4)). In comparison with Table 6, the economy represented here does significantly better overall. That is, having many copies of ATTac-2001 in the economy does not cause them to suffer. However, in this economy, EarlyBidder is able to invade. It gets a significantly higher utility for its clients and only pays slightly more than the ATTac-2001 agents (as computed by utility minus score)[8].

The results in this section suggest that the variance of the closing prices is the largest determining factor between the effectiveness of the two strategies (assuming nobody else is using the open-loop strategy). We speculate that with large price variances, the closed-loop strategy (ATTac-2001) should do better, but with small price variances, the open-loop strategy could do better. The dynamics of TAC-01 seem to have been

[7] EarlyBidderis somewhat reminiscent of the high-bidder agent in [19] in that it commits to an itinerary and bids high enough to ensure its purchase. For expeiments involving 2–6 copies of such bidders, see [19].

[8] We suspect that were the agents allowed to retrain over the course of the experiments, ATTac-2001 would end up improving as occurred in Phase III of the previous set of experiments. Were this to occur, it's possible that EarlyBidderwould no longer be able to invade.

somewhere in the middle ground. Both livingagents and ATTac-2001 had some good games and some bad games, with their eventual scores being almost identical.

## 6 Related and Future Work

Although there has been a good deal of research on auction theory, especially from the perspective of auction mechanisms [11], studies of autonomous bidding agents and their interactions are relatively few and recent. TAC is one example. FM97.6 is another auction test-bed, which is based on fishmarket auctions [14]. Automatic bidding agents have also been created in this domain [6]. There have been a number of studies of agents bidding for a single good in multiple auctions [10, 1, 12].

TAC-01 was the second iteration of the Trading Agent Competition. The rules of TAC-01 are largely identical to those of TAC-00, with three important exceptions:

1. In TAC-00, flight prices did not tend to increase;
2. In TAC-00, hotel auctions usually all closed at the end of the game;
3. In TAC-00, entertainment tickets were distributed uniformly to all agents

While minor on the surface, these changes significantly enriched the strategic complexity of the game. Nonetheless, the agent strategies put forth in TAC-00 [18] were important precursors to the second year's field, for instance as pointed out in Section 4.1.

Having demonstrated ATTac-2001's success at bidding in simultaneous auctions for multiple interacting goods in the TAC-01 domain, our ongoing research agenda includes applying our approach to other similar domains. We particularly expect the boosting approach to price prediction and the decision-theoretic reasoning over price distributions to transfer to other domains.

One candidate domain is the U.S. Federal Communications Commission (FCC) spectrum auctions [20]. Indeed, we have already begun exploring agent strategies in this domain [3, 13]. Other candidate real-world domains include electricity auctions and perhaps even travel booking on public e-commerce sites.

## Acknowledgments

This work was partially supported by the United States-Israel Binational Science Foundation (BSF), grant number 1999038. Thanks to the TAC team at the University of Michigan for providing the infrastructure and support required to run our experiments.

## References

1. Patricia Anthony, Wendy Hall, Viet Dung Dang, and Nicholas R. Jennings. Autonomous agents for participating in multiple on-line auctions. In *Proceedings of the IJCAI-2001 Workshop on E-Business and the Intelligent Web*, Seattle, WA, 2001.
2. Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, 48(1/2/3), 2002.

3. János A. Csirik, Michael L. Littman, Satinder Singh, and Peter Stone. FAucS: An FCC spectrum auction simulator for autonomous bidding agents. In Ludger Fiege, Gero Mühl, and Uwe Wilhelm, editors, *Electronic Commerce: Proceedings of the Second International Workshop*, pages 139–151, Heidelberg, Germany, 2001. Springer Verlag.
4. Clemens Fritschi and Klaus Dorer. Agent-oriented software engineering for successful TAC participation. In *First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Bologna, 2002.
5. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Freeman, San Francisco, CA, 1979.
6. Eduard Gimenez-Funes, Lluis Godo, Juan A. Rodriguez-Aguiolar, and Pere Garcia-Calves. Designing bidding strategies for trading agents in electronic auctions. In *Proceedings of the Third International Conference on Multi-Agent Systems*, pages 136–143, 1998.
7. Amy Greenwald and Justing Boyan. Bidding algorithms for simultaneous auctions. In *Proceedings of Third ACM Conference on E-Commerce*, pages 115–124, Tampa, FL, 2001.
8. J. Harsanyi. Games with incomplete information played by bayesian players. *Management Science*, 14:159–182,320–334,486–502, 1967–1968.
9. Minghua He and Nicholas R. Jennings. SouthamptonTAC: Designing a successful trading agent. In *Fifteenth European Conference on Artificial Intelligence*, Lyon, France, 2002.
10. T. Ito, N. Fukuta, Toramatsu Shintani, and Katia Sycara. Biddingbot: a multiagent support system for cooperative bidding in multiple auctions. In *Proceedings of the Fourth International Conference on MultiAgent Systems*, pages 399–400, July 2000.
11. Paul Klemperer. Auction theory: A guide to the literature. *Journal of Economic Surveys*, 13(3):227–86, July 1999.
12. Chris Preist, Claudio Bartolini, and Ivan Phillips. Algorithm design for agents which participate in multiple simultaneous auctions. In *Agent Mediated Electronic Commerce III (LNAI)*, pages 139–154. Springer-Verlag, Berlin, 2001.
13. Paul S. A. Reitsma, Peter Stone, János A. Csirik, and Michael L. Littman. Self-enforcing strategic demand reduction. In *Workshop on Agent Mediated Electronic Commerce IV: Designing Mechanisms and Systems*, Bologna, Italy, 2002.
14. Juan A. Rodriguez-Aguilar, Francisco J. Martin, Pablo Noriega, Pere Garcia, and Carles Sierra. Towards a test-bed for trading agents in electronic auction markets. *AI Communications*, 11(1):5–19, 1998.
15. Michael H. Rothkopf and Ronald M. Harstad. Modeling competitive bidding: A critical essay. *Management Science*, 40(3):364–384, 1994.
16. Robert E. Schapire and Yoram Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, May/June 2000.
17. Robert E. Schapire, Peter Stone, David McAllester, Michael L. Littman, and János A. Csirik. Modeling auction price uncertainty using boosting-based conditional density estimation. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002. To appear. Available at `http://www.cs.cmu.edu/~pstone/papers.html`.
18. Peter Stone and Amy Greenwald. The first international trading agent competition: Autonomous bidding agents. *Electronic Commerce Research*, 2002. To appear.
19. Peter Stone, Michael L. Littman, Satinder Singh, and Michael Kearns. ATTac-2000: An adaptive autonomous bidding agent. *Journal of Artificial Intelligence Research*, 15:189–206, June 2001.
20. Robert J. Weber. Making more from less: Strategic demand reduction in the FCC spectrum auctions. *Journal of Economics and Management Strategy*, 6(3):529–548, 1997.
21. Michael P. Wellman, Peter R. Wurman, Kevin O'Malley, Roshan Bangera, Shou-de Lin, Daniel Reeves, and William E. Walsh. A trading agent competition. *IEEE Internet Computing*, 5(2):43–51, March/April 2001.