ATTac-2001: A Learning, Autonomous Bidding Agent

Peter Stone, Robert E. Schapire, János A. Csirik, Michael L. Littman, David McAllester
AT&T Labs — Research, 180 Park Avenue, Florham Park, NJ 07932
{pstone,schapire,janos,mlittman,dmac}@research.att.com

Abstract

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.

Introduction

Auctions are becoming an increasingly popular method for transacting business, especially over the internet. When bidding for multiple interacting goods in simultaneous auctions, on the other hand, agents must be able to reason about uncertainty and make complex value assessments. For example, an agent bidding on one's behalf for a camera and 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 some of the details of the agent as well as underlying principles that we believe have applications in a wide variety of bidding situations.

ATTac-2001 must bid simultaneously for multiple interacting goods. We start with the observation that the key challenge is the modeling of uncertainty in the 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 uncertainty and, to the greatest extent possible, analytically calculate optimal bids. ATTac-2001 uses a predictive, data-driven approach to bidding based on 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 over the course of the competition.

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

General Approach

In a wide variety of decision-theretic 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. We believe that profit-prediction for hypothetical situations can be a useful tool in a wide variety of situations.

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 predicted prices. The predicted prices can, of course, be different in different situations, e.g., previous closing prices can be relevant to predicting future closing prices. We believe that price prediction is one of the central problems to be faced in a wide variety of bidding situations.

A more sophisticated approach to profit-prediction, used in some versions of our agent, is to construct a model of the probability distribution over 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 hypothesitcal future situations.

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

[2]The problem appears computationally difficult in general, but was solved effectively in TAC-00 (Greenwald & Boyan 2001; Stone *et al.* 2001).

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 (Wellman *et al.* 2001), 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.

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. 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.

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, four teams from each group were selected to compete in the finals during that same afternoon.

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.

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 $v(G^*)$; this can be found via integer linear programming (Stone *et al.* 2001).

We found that the integer linear programs we encountered in TAC-01 generally required less than 0.01 seconds on a 650 MHz Pentium II using the "LPsolve" package. However, there were many instances where the ILP solver took significantly longer, so we opted to use a bound from the much more consistent linear programming relaxation when only $v(G^*)$was needed (as opposed to G^*itself).

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.

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;[3] 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.

Space limitations preclude more than a terse description of the supervised learning algorithm that we used. Very briefly, 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 (2000) combined with a modification of boosting algorithms for logistic regression proposed by Collins, Schapire and Singer (2002).

Cost of Additional Rooms

Our hotel price predictor described above assumes that ATTac-2001's bids do not affect the ultimate closing price. Because the economy is so small, we decided to modify the prices sent to the allocator to discourage overbidding for any single room. We did this by inflating predicted prices for three or more rooms of the same type by a multiplicative factor (1.35) determined from an analysis of data from previous games. Using the modified prices in the allocator tended to spread out ATTac-2001's demand over the different hotel auctions.

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 room that is still open, ATTac-2001 assumes that that hotel will close next and computes the marginal value of that hotel room given the predicted closing prices of the other hotel rooms. If the hotel 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 for computation between closing times (or 30 seconds if there are still flights to consider), ATTac-2001 divides the available time among the different open hotel

[3] 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.

rooms 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 hotels.

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 ith 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 ith 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.

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. ATTac-2001 took the cost of postponing commitment to be the average predicted cost of the flight (computed as above) over the next *flight-lookahead* whole minutes. 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 detemining the marginal value of hotel rooms.

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 less 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.

Results

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. The finals consisted of 24 games among the same eight agents. In raw score, ATTac-2001 ended up finishing a very close second to livingagents (Fritschi & Dorer 2002), scoring an average of two points less per game.

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 agents' 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.

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.

Here 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 placed bids for these goods at sufficiently high prices to ensure that they would be purchased right at after the first flight quotes. It then 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, which consisted 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.

Agent	*Relative Score*	
	Phase I	*Phase III*
ATTac-2001$_{ns}$	105.2 ± 49.5 (2)	166.2 ± 20.8 (1)
ATTac-2001$_s$	27.8 ± 42.1 (3)	122.3 ± 19.4 (2)
EarlyBidder	140.3 ± 38.6 (1)	117.0 ± 18.0 (3)
SimpleMean$_{ns}$	-28.8 ± 45.1 (5)	-11.5 ± 21.7 (4)
SimpleMean$_s$	-72.0 ± 47.5 (7)	-44.1 ± 18.2 (5)
Cond'lMean$_{ns}$	8.6 ± 41.2 (4)	-60.1 ± 19.7 (6)
Cond'lMean$_s$	-147.5 ± 35.6 (8)	-91.1 ± 17.6 (7)
CurrentBid	-33.7 ± 52.4 (6)	-198.8 ± 26.0 (8)

Table 3: The average relative scores for eight agents in the first and last of 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 3 shows how the agents performed in the first and last 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 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.

We were also surprised that ConditionalMean$_s$ and ConditionalMean$_{ns}$ eventually performed worse than the less sophisticated SimpleMean$_s$ and SimpleMean$_{ns}$. We do not fully understand why this happened. One possibility is that the simpler model happens to give predictions that are just as good as the more complicated model, perhaps be-

cause 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).

As a measure of the inaccuracy of the predictions made by the three non-sampling agents, we measured the root mean squared error of the predicitons 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 4 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.

Agent	Score	Utility
ATTac-2001	2431 $\pm$ 464	8909 $\pm$ 264
EarlyBidder(7)	-4880 $\pm$ 337	9870 $\pm$ 34

Table 4: The results of running ATTac-2001 against 7 copies of EarlyBidder over the course of 27 games.

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 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 5 shows the results of 7 copies of ATTac-2001 playing against each other and one copy of the EarlyBidder. 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.

From the results in Table 5 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 4, 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).

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

Table 5: 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.

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. In the end, the dynamics of the TAC-01 competition seem to have been 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.

Conclusion

ATTac-2001 is a learning autonomous bidding agent successfully designed to compete in TAC-01, a domain featuring simultaneous auctions for multiple interacting goods. Our future research agenda includes applying our approach to other similar domains, such as the U.S. Federal Communications Commition (FCC) spectrum auctions (Weber 1997).

References

Collins, M.; Schapire, R. E.; and Singer, Y. 2002. Logistic regression, AdaBoost and Bregman distances. *Machine Learning* 48(1/2/3).

Fritschi, C., and Dorer, K. 2002. Agent-oriented software engineering for successful TAC participation. In *First International Joint Conference on Autonomous Agents and Multi-Agent Systems*.

Greenwald, A., and Boyan, J. 2001. Bidding algorithms for simultaneous auctions. In *Proceedings of Third ACM Conference on E-Commerce*, 115–124.

Schapire, R. E., and Singer, Y. 2000. BoosTexter: A boosting-based system for text categorization. *Machine Learning* 39(2/3):135–168.

Stone, P.; Littman, M. L.; Singh, S.; and Kearns, M. 2001. ATTac-2000: An adaptive autonomous bidding agent. *Journal of Artificial Intelligence Research* 15:189–206.

Weber, R. J. 1997. Making more from less: Strategic demand reduction in the FCC spectrum auctions. *Journal of Economics and Management Strategy* 6(3):529–548.

Wellman, M. P.; Wurman, P. R.; O'Malley, K.; Bangera, R.; Lin, S.-d.; Reeves, D.; and Walsh, W. E. 2001. A trading agent competition. *IEEE Internet Computing* 5(2):43–51.