WhiteBear: An Empirical Study of Design Tradeoffs for Autonomous Trading Agents

Ioannis A. Vetsikas and **Bart Selman**
Department of Computer Science
Cornell University
Ithaca, NY 14853
{vetsikas,selman}@cs.cornell.edu

Abstract

This paper presents WhiteBear, one of the top-scoring agents in the 2^{nd} International Trading Agent Competition (TAC). TAC was designed as a realistic complex test-bed for designing agents trading in e-marketplaces. Our architecture is an adaptive, robust agent architecture combining principled methods and empirical knowledge. The agent faced several technical challenges. Deciding the optimal quantities to buy and sell, the desired prices and the time of bid placement was only part of its design. Other important issues that we resolved were balancing the aggressiveness of the agent's bids against the cost of obtaining increased flexibility and the integration of domain specific knowledge with general agent design techniques. We present our observations and back our conclusions with empirical results.

Introduction

The Trading Agent Competition (TAC) was designed and organized by a group of researchers at the University of Michigan, led by Michael Wellman (Wellman *et al.* 2001). In earlier work, e.g. (Anthony *et al.* 2001), (Greenwald, Kephart, & Tesauro 1999), (Preist, Bartolini, & Phillips 2001), researchers tested their ideas about agent design in smaller market games that they designed themselves. Time was spent on the design and implementation of the market, but there was no common market scenario that researchers could focus on and use to compare strategies. TAC provides such a common framework. It is a challenging benchmark domain which incorporates several elements found in real marketplaces in the realistic setup of travel agents that organize trips for their clients. It includes several complementary and substitutable goods[1] traded in a variety of auctions by autonomous agents that seek to maximize their profit while minimizing expenses. These agents must decide the bids to be placed. The problem is isomorphic to variants of the winner determination problem in combinatorial auctions (Greenwald & Boyan 2001). However, instead of bidding for bundles and letting the auctioneer determine the final allocation that maximizes income (for this problem see (Fujishima, Leyton-Brown, & Shoham 1999), (Sandholm & Suri 2000)), in this setting the computational cost is moved to the agents who have to deal with complementarities and substitutabilities of goods.

Participating in TAC gave us the opportunity to experiment with general agent design methodologies. We study design tradeoffs applicable to most market domains. We examine the tradeoff of the agent paying a premium either for acquiring information or for having a wider range of actions available against the increased flexibility that this information provides. In addition, using the information known about the individual auctions and modelling the prices helps the agent to judge its flexibility more accurately and allows it to strike a balance between the cost of having an increased flexibility and the benefit obtained from it, which in turn improves overall performance even more. We also examine how agents of varying degrees of bidding aggressiveness perform in a range of environments against other agents of different aggressiveness. We find that there is a certain optimal level of "aggressiveness": an agent that is just aggressive enough to implement its plan outperforms agents who are either not aggressive enough or too aggressive. As we will see, the resulting agent strategy is quite robust and performs well in a range of agent environments.

We also show that even though generating a good plan is crucial for the agent to maximize its utility, it is not necessary to compute the optimal plan. We will see that a randomized greedy search methods produces plans that are very close to optimal and more than good enough for effective bidding. In fact, most of the time the agent has only rather incomplete knowledge of the various price levels (auctions are still open), therefore solving the optimization task optimally does not necessarily lead to better overall performance. Our results show that it is actually more important to be able to quickly update the current plan whenever new information comes in. Our fast planning strategy allows us to do so (plan generation generally in less than 1 second).

One substantial benefit of our modular agent architecture is that it is also able to combine seamlessly both principled methods and methods based on empirical knowledge. Overall our agent is adaptive, versatile, fast and robust and its elements are general enough to work well under any situation that requires bidding in multiple simultaneous auctions. We have demonstrated this not only in the controlled experi-

[1]The various entertainment tickets are substitutable, while hotels, plane and entertainment tickets are complementary goods.

ments but also in practice by performing consistently among the top scoring agents in the TAC.

The paper is organized as follows. In the next section, we describe the TAC market game formally. After that we present our agent design architecture. In the sections afterwards we present the results of the competition and we explain the experiments performed to validate our conclusions. Finally we discuss possible directions for future work and conclude.

TAC: A description of the Market Game

In the Trading Agent Competition, an autonomous trading agent competes against 7 other agents in each game. Each agent is a travel agent with the goal of arranging a trip to Tampa for $CUST$ customers. To do so it must purchase plane tickets, hotel rooms and entertainment tickets. This is done in simultaneous online auctions. The agents send bids to the central server running at the University of Michigan and are informed about price quotes and transaction information. Each type of commodity (tickets, rooms) is sold in separate auctions (for a survey of auction theory see (Klemperer 1999)). Each game lasts for 12 minutes (720 seconds).

There is only one flight per day each way with an infinite amount of tickets, which are sold in separate, continuously clearing auctions in which prices follow a random walk. For each flight a hidden parameter x is chosen uniformly in $[10, 90]$. The initial price is chosen uniformly in $[250, 400]$ and is perturbed by a value drawn uniformly in $[-10, 10 + \frac{t}{720} \cdot (x - 10)]$, where t (*sec*) is the time elapsed since the game started. The agents may not resell tickets.

There are two hotels in Tampa: the Tampa Towers (cleaner and more convenient) and the Shoreline Shanties (the not so good and expected to be cheaper hotel). There are 16 rooms available each night at each hotel. Rooms for each of the days are sold by each hotel in separate, ascending, open-cry, multi-unit, 16^{th}-price auctions. A customer must be given a hotel room for every night between his arrival and the night before his departing date and they are not allowed to change hotel during their stay. These auctions close at randomly determined times and more specifically at minute 4 (4:00) a randomly selected auction closes, then another at 5:00, until at 11:00 the last one closes. No prior knowledge of the closing order exists and agents may not resell rooms or bids.

Entertainment tickets are traded (bought and sold) among the agents in continuous double auctions (stock market type auctions) that close when the game ends. Bids match as soon as possible. Each agent starts with an endowment of 12 random tickets and these are the only tickets available in the game.

Each customer i has a preferred arrival date PR_i^{arr} and a preferred departure date PR_i^{dep}. She also has a preference for staying at the good hotel represented by a utility bonus UH_i as well as individual preferences for each entertainment event j represented by utility bonuses $UENT_{i,j}$. Let $DAYS$ be the total number of days and ET the number of different entertainment types.

The parameters of customer i's itinerary that an agent has to decide upon are the assigned arrival and departure dates, AA_i and AD_i respectively, whether the customer is placed in the good hotel GH_i (which takes value 1 if she is placed in the Towers and 0 otherwise) and $ENT_{i,j}$ which is the day that a ticket of the event j is assigned to customer i (this is e.g. 0 if no such ticket is assigned).

The utility that the travel plan has for each customer i is:

$$\begin{aligned} util_i &= 1000 + UH_i \cdot GH_i \\ &+ \sum_{d=AA_i}^{AD_i} \max_j \Big\{ UENT_{i,j} \cdot \mathbf{I}(ENT_{i,j} = d) \Big\} \\ &- 100 \cdot \Big(|PR_i^{arr} - AA_i| + |PR_i^{dep} - AD_i| \Big) \end{aligned} \tag{1}$$

if $1 \leq AA_i < AD_i \leq DAYS$,
else $util_i = 0$, because the plan is not feasible.

It should be noted that only one entertainment ticket can be assigned each day and this is modeled by taking the maximum utility from each entertainment type on each day. We assume that an unfeasible plan means no plan (e.g. $AA_i = AD_i = 0$). The function $\mathbf{I}(bool\_expr)$ is 1 if the $bool_expr =$TRUE and 0 otherwise.

The total income for an agent is equal to the sum of its clients' utilities. Each agents searches for a set of itineraries (represented by the parameters AA_i, AD_i, GH_i and $ENT_{i,j}$) that maximize this profit while minimizing its expenses.

The WhiteBear Architecture

The optimization problem in the 2^{nd} TAC is the same as the one in the first one, so we knew strategies that worked well in that setting and had some idea of where to start in our implementation (Greenwald & Stone 2001). However, the new auction rules were different enough to introduce several issues that markedly changed the game and made it impossible to import agents without making extensive modifications. For an agent architecture to be useful in a general setting, it must be *adaptive*, *flexible* and *easily modifiable*, so that it is possible to make modifications on the fly and adapt the agent to the system in which it is operating. These were lessons that we incorporated in to the design of our architecture.[2]

In addition, as information is gained by participation in the system, the agent architecture must allow and *facilitate the incorporation of the knowledge obtained*. During the competition we changed many parts of our agent to reflect the knowledge that we obtained. For example, we initially experimented with bidding based mainly on heuristics, but it did not take us too long to realize that this approach was not able to adapt fast enough to the swift pace required by bidding in simultaneous auctions. Bidding without an overall plan is quite inefficient, because of the complementarities and substitutabilities between goods. We therefore decided to formulate and solve the optimization problem of maximizing the utility of our agent (see Planner section). Such

[2]The original architecture proposed by the TAC team seemed an appropriate starting point (with a number of necessary modifications).

a fundamental change would be difficult and quite time-consuming, if our architecture did not support interchangeable parts.

The agent architecture is summarized as follows:

while (not end of game) {
1. Get price quotes and transaction information
2. Calculate price estimates
3. *Planner*: Form and solve optimization problem
4. *Bidder*: Bid according to plan and price estimates

}

Our architecture is highly modular and each component of the agent can be replaced by another or modified. In fact several parts of the components themselves are also substitutable. This highly modular and substitutable architecture allowed us to test several different strategies and to run controlled experiments as well. Also note our agent repeatedly reevaluates its plan and bids accordingly, since in most domains it is crucial to react fast to information on the domain and to other agents actions and the TAC game is no exception to this rule. We furthermore designed our components to be as fast and adaptive as possible without sacrificing efficiency.

In the next subsection we describe how the agent generates the price estimate vectors. In the subsection afterwards we explain how the agent forms and then solves the optimization problem of maximizing its utility. In the last subsection we present the bidding strategies used by the bidder to implement the generated plan.

Price Estimate Vectors

In order to formulate the optimization problem that the planner solves, it is necessary to estimate the prices at which commodities are expected to be bought or sold. We started from the "priceline" idea presented in (Greenwald & Boyan 2001) and we simplified and extended it where appropriate. We implemented a module which calculates *price estimate vectors* (PEV). These contain the value (price) of the x^{th} unit for each commodity. For some goods this price is the same for all units, but for others it is not; e.g. buying more hotel rooms usually increases the price one has to pay.

Let O_d^{arr}, O_d^{dep}, O_d^{goodh}, O_d^{badh}, $O_{d,t}^{ent}$ and $PEV_d^{arr}(x)$, $PEV_d^{dep}(x)$, $PEV_d^{goodh}(x)$, $PEV_d^{badh}(x)$, $PEV_{d,t}^{ent}(x)$ be respectively the quantities owned and the the price estimates of the x^{th} unit for the plane tickets, the hotel rooms and the entertainment tickets for event t for each day d. Let us also define the operator $\sigma(f(x), z) = \sum_{i=1}^{z} f(i)$.

The price estimate vectors are

$$PEV_{d,t}^{ent}(x) \begin{cases} \leq \text{current bid price} & \text{if } x \leq O_{d,t}^{ent} \\ \geq \text{current ask price} & \text{if } x > O_{d,t}^{ent} \end{cases} \tag{2}$$

$$PEV_d^{type}(x) \begin{cases} = 0 & \text{if } x \leq O_d^{type} \\ \geq \text{current ask price} & \text{if } x > O_d^{type} \end{cases} \tag{3}$$

where *type*$\in\{$*arr,dep,goodh,badh*$\}$.

Since once bought an agent cannot sell plane tickets nor hotel rooms to anyone else, this means that their value is 0 for the agent; it has to treat the money that it has already paid for them as a "sunk cost". The agent must buy all rooms that it is currently winning if the auction closes, so these are also considered owned by the planner. For the plane tickets the PEV's are equal to the *current ask price*, for tickets not yet bought, but for hotel rooms it is higher than that. In fact, we estimate the price of the next room bought as a function of the current price and for each additional room the price estimate increases by an amount proportional to the current price and the day and type the room (rooms on days 2 and 3 or on the good hotel are estimated to cost more than rooms on days 1 and 4 or on the bad hotel). We also set $PEV_{d,t}^{ent}(x) = 0, \text{if } x < O_{d,t}^{ent}$ and $PEV_{d,t}^{ent}(x) > 200, \text{if } x > O_{d,t}^{ent} + 1$, because all we are certain from observing the current ask and bid prices is that there is at least one ticket up for buy and sale, but we're not sure if there are any more. Also note that entertainment tickets owned by the agent do not have a value of 0, since they can be sold. By using them, the agent loses a value equal to the price at which it could have sold them. Therefore the PEV of owned tickets is equal to the price at which they are expected to be sold.

Planner

The planner is another module in our architecture. It uses the PEV's to generate the customers itineraries that maximize the agent's utility. Therefore, it also provides the type and total quantity of commodities that should be purchased or sold to achieve this. In order to do this the planner formulates the following optimization problem:

$$\max_{AA_c, AD_c, GH_c, ENT_{c,t}} \left\{ \sum_{c=1}^{CUST} util_c - COST \right\} \tag{4}$$

where the cost of buying the resources is

$$\begin{aligned} COST = \sum_{d=1}^{DAYS} \Big[&\sigma\Big(PEV_d^{arr}(x), \sum_{c=1}^{CUST} \mathbf{I}(AA_c = d)\Big) \\ &+ \sigma\Big(PEV_d^{dep}(x), \sum_{c=1}^{CUST} \mathbf{I}(AD_c = d)\Big) \\ &+ \sigma\Big(PEV_d^{goodh}(x), \sum_{c=1}^{CUST} [GH_c \cdot \mathbf{I}(AA_c \leq d < AD_c)]\Big) \\ &+ \sigma\Big(PEV_d^{badh}(x), \sum_{c=1}^{CUST} [(1 - GH_c) \cdot \mathbf{I}(AA_c \leq d < AD_c)]\Big) \\ &+ \sum_{t=1}^{ET} \sigma\Big(PEV_{d,t}^{ent}(x), \sum_{c=1}^{CUST} \mathbf{I}(ENT_{c,t} = d)\Big) \Big] \end{aligned} \tag{5}$$

Once the problem has been formulated, the next step is solving it. This problem is NP-complete, but for the size of the TAC problem an optimal solution usually can be produced fast. In order to create a more general algorithm we realized that it should scale well with the size of the problem and should not include elaborate heuristics applicable only to the TAC problem. Thus we chose to implement a greedy algorithm: the order of customers is randomized and then each customer's utility is optimized separately. This is done a few hundred times in order to maximize the chances that the solution will be optimal most of the time. In practice we

have found the following additions to be quite useful:
(i) Compute the utility of the plan $\mathcal{P}_1$ from the previous loop before considering other plans. Thus the algorithm always finds a plan $\mathcal{P}_2$ that is at least as good as $\mathcal{P}_1$ and there are relatively few radical changes in plans between loops. We observed empirically that this prevented some radical bid changes and improved efficiency.
(ii) We added a constraint that dispersed the bids of the agent for resources in limited quantities (hotel rooms in TAC). Plans, which demanded for a single day more than 4 rooms in the same hotel, or more than 6 rooms in total, were not considered. This leads to some utility loss in rare cases. However, bidding heavily for one room type means that overall demand will very likely be high and therefore prices will skyrocket, which in turn will lower the agent's score significantly. We observed empirically that the utility loss from not obtaining the best plan tends to be quite small compared to the expected utility loss from rising prices.

We have also verified that this randomized greedy algorithm gives solutions which are often optimal and never far from optimal. We checked the plans (at the game's end) that were produced by 100 randomly select runs and observed that over half of the plans were optimal and on average the utility loss was about 15 points (out of 9800 to 10000 usually[3]), namely close to 0.15%. Compared to the usual utility of 2000 to 3000 that our agents score in most games, they achieved about 99.3% to 99.5% of optimal. These observations are consistent with the ones about a related greedy strategy in (Stone *et al.* 2001). Considering that at the beginning of the game the optimization problem is based on inaccurate values, since the closing hotel prices are not known, an 100%-optimal solution is not necessary and can be replaced by our almost optimal approximation. As commodities are bought and the prices approach their closing values, most of the commodities needed are already bought and we have observed empirically that bidding is rarely affected by the generation of approximately optimal solutions instead of optimal ones.

This algorithm takes approximately *1 second* to run through 500 different randomized orders and compute an allocation for each. Our test bed was a cluster of 8 Pentium III 550 MHz CPU's, with each agent using no more than one cpu. This system was used for all our experiments and our participation in the TAC.[4] Hence it is verified that our goal to provide a planner that is fast and not domain specific is accomplished and not at the expense of the overall agent performance.

Bidding Strategies

Once the plan has been generated the bidder places separate bids for all the commodities needed to implement the plan. During the competition every team, including ours, used empirical observations from the games it participated in (over 1000) in order to improve its strategy. We experimented with several different approaches. In the next sections we describe the possible bidding strategies for the different goods and the tradeoffs that we faced.

Paying for adaptability The purchase of flight tickets presents a very interesting dilemma. We have verified that ticket prices are expected to increase approximately in proportion to the square of the time elapsed since the start of the game. This means that the more one waits the higher the prices will get and the increase is more dramatic towards the end of the game. From that point of view, if an agent knows accurately the plan that it wishes to implement, it should buy the plane tickets immediately. On the other hand, if the plan is not known accurately (which is usually the case), the agent should wait until the prices for hotel rooms have been determined. This is because buying plane tickets early restricts the flexibility (adaptability) that the agent has in forming plans: e.g. if some hotel room that the agent needs becomes too expensive, then if it has already bought the corresponding plane tickets, it must either waste these, or pay a high price to get the room. An obvious tradeoff exists in this case, since delaying the purchase of plane tickets increases the flexibility of the agent and hence provides the potential for a higher income at the expense of some monetary penalty.

Our initial attempt was similar to some of our competitors (as we found out later, during the finals): we decided to bid for the tickets between 4:00 and 5:00. This is because most agents bid for the hotel rooms that they need right before 4:00, therefore after this time the room prices approximate sufficiently their closing prices and a plan created at that time is usually similar to the optimal plan for known closing prices. We tried waiting for a later minute, but we empirically observed that the price increase was more than the benefit from waiting. A further improvement is to buy some tickets at the start of the game. These tickets are bought based on the prices and the preferences of the customers (preference is given to tickets on days 1 and 5, to cheaper tickets and to customers with shorter itineraries). About 50% of the tickets are bought in this way and we have observed that these tickets are rarely wasted. Another improvement is to bid for 2 less tickets per flight than required by the plan until 4:00 and for 1 less until 5:00; this provides greater flexibility and it is highly unlikely that the final optimal plan will not make use of these tickets.

A further improvement was obtained by estimating the hidden parameter x of each flight. Let $\{y_i, t_i\}, \forall i \in \{1, \ldots, N\}$ be the pairs of the price changes y_i observed at times t_i and let N be the number of such pairs. Let $A = x - 10$. Then $A \in \{0, \ldots, 80\}$ and $P[A = z] = \frac{1}{81}, \forall z \in \{0, \ldots, 80\}$ and $P[A = z] = 0, \forall z \notin \{0, \ldots, 80\}$ since x is uniformly chosen among the integers in $[10, 90]$. Since A is independent of the times $\{t_i = T_i\}$ when the changes occur, therefore $P[A = z] = P[A = z | \{\bigwedge_{i=1}^{N} (t_i = T_i)\}]$. Also y_i is uniformly chosen in $\{-10, \ldots, 10 + \lfloor \frac{A \cdot T_i}{720} \rfloor\}$, therefore $P[y_i = Y_i | (A = z) \wedge (t_i = T_i)] = \frac{1}{\lfloor \frac{A \cdot T_i}{720} \rfloor + 21}$, if Y_i is an integer in $[-10, 10 + \frac{A \cdot t_i}{720}]$ and $P[y_i = Y_i | (A = z) \wedge (t_i = T_i)] = 0$, otherwise. In addition the price change y_i only de-

[3]These were the scores of the allocation at the end of the game (no expenses were considered).

[4]During the competition only one processor was used, but during the experimentations we used all 8, since 8 different instantiations of the agent were running at the same time.

pends on time t_i and is independent of all $t_j, \forall j \neq i$, therefore $P[y_i = Y_i | (A = z) \wedge (t_i = T_i)] = P[y_i = Y_i | (A = z) \wedge \{\bigwedge_{i=1}^{N} (t_i = T_i)\}]$. Given the observations made so far, the probability that $A = z$ for any $z \in \{0, \ldots, 80\}$ is

$$P[A = z | \{\bigwedge_{i=1}^{N} (t_i = T_i)\} \wedge \{\bigwedge_{i=1}^{N} (y_i = Y_i)\}] = \frac{P[(A=z) \wedge \{\bigwedge_{i=1}^{N} (y_i=Y_i)\} | \bigwedge_{i=1}^{N} (t_i=T_i)]}{P[\bigwedge_{i=1}^{N} (y_i=Y_i) | \bigwedge_{i=1}^{N} (t_i=T_i)]} = \frac{P[A=z | \bigwedge_{i=1}^{N} (t_i=T_i)] \cdot p_z}{\sum_{\zeta=0}^{80} \left\{ P[A=\zeta | \bigwedge_{i=1}^{N} (t_i=T_i)] \cdot p_\zeta \right\}} = \frac{\frac{1}{81} \cdot p_z}{\sum_{\zeta=0}^{80} \frac{1}{81} \cdot p_\zeta} = \frac{p_z}{\sum_{\zeta=0}^{80} p_\zeta},$$

where $p_z = P[\bigwedge_{i=1}^{N} (y_i = Y_i) | (A = z) \wedge \{\bigwedge_{i=1}^{N} (t_i = T_i)\}]$. Given the value of A, $\{y_i = Y_i\}$ are independent of each other, thus $p_z = \prod_{i=1}^{N} P[y_i = Y_i | (A = z) \wedge \{\bigwedge_{i=1}^{N} (t_i = T_i)\}] = \prod_{i=1}^{N} P[y_i = Y_i | (A = z) \wedge (t_i = T_i)]$. Since p_z is a product, that means that if any of conditional probabilities who make up the product is zero then $p_z = 0$ as well. Therefore for $p_z \neq 0$ it must be that $\forall i, -10 \leq Y_i \leq 10 + \frac{A \cdot T_i}{720} \Leftrightarrow \forall i, A \geq \frac{720 \cdot (Y_i - 10)}{T_i} \Leftrightarrow A \geq \max_i \frac{720 \cdot (Y_i - 10)}{T_i}$. Therefore $p_z = \prod_{i=1}^{N} \frac{1}{\lfloor \frac{A \cdot T_i}{720} \rfloor + 21}$, if $A \geq \max_i \frac{720 \cdot (Y_i - 10)}{T_i}$ and $p_z = 0$, otherwise. So we conclude that

$$P[x = z | \{\bigwedge_{i=1}^{N} (t_i = T_i)\} \wedge \{\bigwedge_{i=1}^{N} (y_i = Y_i)\}] = \frac{p_z}{\sum_{\zeta=10}^{90} p_\zeta},$$

where $p_z = \prod_{i=1}^{N} \frac{1}{\lfloor \frac{(z-10) \cdot T_i}{720} \rfloor + 21}$, if $z \geq \max_i \frac{720 \cdot (Y_i - 10)}{T_i} + 10$ and $z \leq 90$. Otherwise $p_z = 0$.

In practice we have observed that just the knowledge of the smallest value of z for which $p_z > 0$ is enough to estimate the price increase and that we obtain this knowledge quite fast (around 2:00 to 3:00 usually). This information is then used to bid earlier for tickets whose price is very likely to increase and to wait more for tickets whose price is expected to increase little or none (they are bought when their usefulness for the agent is almost certain).[5]

Bid Aggressiveness Bidding for hotel rooms poses some interesting questions as well. The main issue in this case seems to be how aggressively each agent should bid (the level of the prices it submits in its bids). Originally we decided to have the agent bid an increment higher than the current price. The agent also bids progressively higher for each consecutive unit of a commodity for which it wants more than one unit. E.g. if the agent wants to buy 3 units of a hotel room, it might bid 210 for the first, 250 for the second and 290 for the third. This is the lowest (L) possible aggressiveness we have tried. Another bidding strategy that we have used is that the agent bids progressively closer to the marginal utility δU as time passes[6]. This is the highest (H) aggressiveness level that we have tried. As a compromise between the two extreme levels we also implemented a version of the agent that bids like the aggressive (H) agent for rooms that have a high value of δU and bids like the non-aggressiveness (L) agent otherwise. This is the agent of medium (M) aggressiveness. For more details into the behavior of these variants we run several experiments and the results are presented later in the paper.

[5]If this procedure is not used, then the agent pays an average extra cost of 240 to 300 because it waits until minutes 4 and 5 to buy the last tickets. If it is used, then this cost is *almost halved.*

[6]The marginal utility δU for a particular hotel room is the change in utility that occurs if the agent fails to acquire it. In fact for each customer i that needs a particular room we bid $\frac{\delta U}{z}$ instead of δU, where z is the number of rooms which are still needed to complete her itinerary. We do this in order not to drive the prices up prematurely.

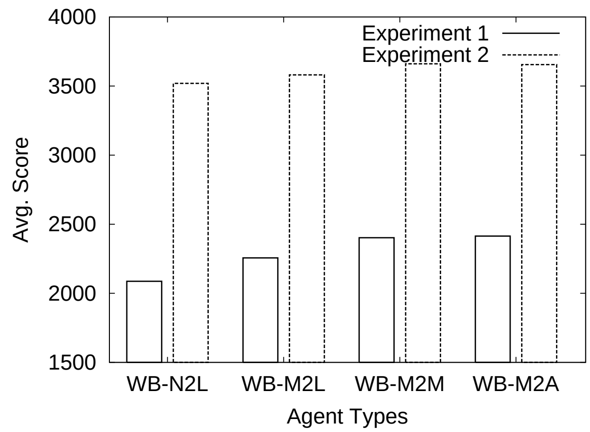

Figure 1: Average scores for some agents that do and some that do not model the flight ticket prices in two experiments

As far as the timing of the bids is concerned, there is little ambiguity about what a very good strategy is. The agent waits until 3:00 (and before 4:00) and then places its first bids based on the plan generated by the planner. The reason for this is that it does not wish to increase the prices earlier than necessary nor to give away information to the other agents. We also observed empirically that an added feature which increases performance is to place bids for a small number of rooms at the beginning of the game at a very low price (whether they are needed or not). In case these rooms are eventually bought, the agent pays only a very small price and gains increased flexibility in implementing its plan.

Entertainment The agent buys (sells) the entertainment tickets that it needs (does not need) for implementing its plan at a price equal to the current price plus (minus) a small increment. The only exceptions to this rule are:
(i) At the game's start and depending on how many tickets the agent begins with, it will offer to buy tickets at low prices, in order to increase its flexibility at a small cost. Even if these tickets are not used the agent often manages to sell them for a profit.
(ii) The agent will not buy (sell) at a high (low) price, even if this is beneficial to its utility, because otherwise it helps other agents. This restriction is somewhat relaxed at 11:00, in order for the agent to improve its score further, but it will still avoid some beneficial deals if these would be very profitable for another agent.

Experimental Results

To verify our observations and show the generality of our conclusions, we decided to perform several controlled experiments. Thus we gain more precise knowledge of the behavior of different types of agents in various situations.

	Agent Scores								Average Scores			Statistically Significant Difference?		
#WB-M2H	1	2	3	4	5	6	7	8	WB-M2L	WB-M2M	WB-M2H	M2L/M2M	M2M/M2H	M2L/M2H
0 (178)	2614	2638	2490	2463	2421	2442	2526	2455	2466	2626	N/A	✓		
2 (242)	2339	2350	2269	2265	2229	2241	2347	2371	2251	2344	2359	✓	×	✓
4 (199)	2130	2073	2072	2029	2046	2098	2048	2033	2051	2101	2056	×	×	×
6 (100)	1112	1165	796	843	920	884	848	898	N/A	1138	865		✓	

Table 2: Scores for agents WB-M2L, WB-M2M and WB-M2H as the number of aggressive agents (WB-M2H) participating increases. In each experiment agents 1 and 2 are instances of WB-M2M. The agents above the stair-step line are WB-M2L, while the ones below are WB-M2H. The averages scores for each agent type are presented in the next rows. In the last rows, ✓indicates statistically significant difference in the scores of the corresponding agents, while × indicates similar scores (statistically significant).

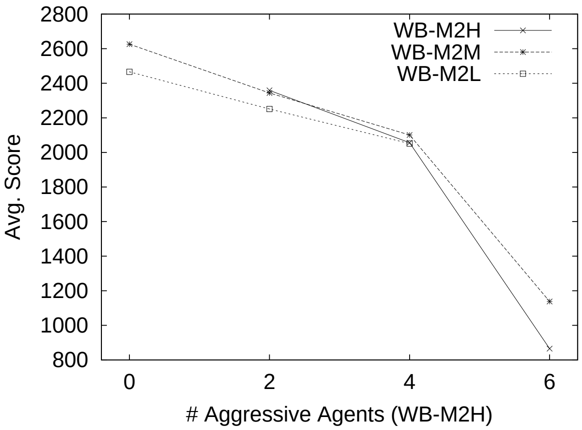

Figure 2: Changes in agent's average scores as the number of very aggressive agents participating in the game increases

Experiments		WB-N2L	WB-M2L	WB-M2M	WB-M2H
Exp 1 (144)	agent 1	2087	2238	2387	2429
	agent 2	2087	2274	2418	2399
	average	2087	2256	2402	2414
Exp 2 (200)		3519	3581	3661	3656

Table 1: Average scores of agents WB-N2L, WB-M2L, WB-M2M and WB-M2H. For experiment 1 the scores of the 2 instances of each agent type are also averaged. *The number inside the parentheses is the total number of games for each experiment and this will be the case for every table.*

To distinguish between the different versions of the agent, we use the notation WB-xyz, where (i) x is M if the agent models the plane ticket prices and N if this feature is not used, (ii) y takes values 0, 1 or 2 and is equal to the maximum number of plane tickets per flight which are not bought before 4:00, in order not to restrict the planner's options[7] and (iii) z characterizes the aggressiveness with which the agent bids for hotel rooms and takes values L,M and H for low, medium and high degree of aggressiveness respectively. To formally evaluate whether an agent type outperforms another type, we use *paired t-tests* for all our experiments; values of less than 10% are considered to indicate a statistically significant difference (in most of the experiments the values are actually well below 5%). If more than one instances of a certain agent type participate in an experiment, we compute the t-test for all possible combinations of instances.[8]

The first set of experiments were aimed at verifying our observation that modeling the plane ticket prices improves the performance of the agent. This is something that we expected, since the agent uses this information to bid later for tickets whose price will not increase much (therefore achieving a greater flexibility at low cost), while bidding early for tickets whose price increases faster (therefore saving considerable amount of money). We run 2 experiments with the following 4 agent types: WB-N2L, WB-M2L, WB-M2M and WB-M2H. In the first we run 2 instances of each agent, while in the second we run only one and the other 4 slots were filled with the standard agent provided by the TAC support team. The result of this experiment are presented in table 1. The other agents, which model the plane ticket prices, perform better than agent WB-N2L, which does not do so (figure 1). The differences between WB-N2L and the other agents are statistically significant, except for the one between WB-N2L and WB-M2L in experiment 2. We also observe that WB-M2L is outperformed by agents WB-M2M and WB-M2H, which in turn achieve similar scores; these results are statistically significant for experiment 1.

Having determined that modeling of plane ticket prices leads to significant improvement, we concentrated our attention to agents WB-M2L, WB-M2M and WB-M2H in the next series of experiments. For all experiments, the number of instances of agent WB-M2M was 2, while the number of aggressive agents WB-M2H was increased from 0 to 6. The rest of the slots were filled with instances of agent WB-M2L. The result of this experiment are presented in table 2. By increasing the number of agents which bid more aggressively, there is more competition between agents and the hotel room prices increase, leading to a decrease in scores. While the number of aggressive agents #WB-M2H$\leq$4, the decrease in score is relatively small for all agents and is approximately linear with #WB-M2H; The aggressive agents (WB-M2H) do relatively better in less competitive environ-

[7] In the bidding section, we mentioned that until 4:00 the agent buys $PL_d^{arr} - y$ and $PL_d^{dep} - y$ plane tickets, where PL_d^{arr}, PL_d^{arr} are the total number of plane tickets needed to implement the plan. In that case we used $y = 2$. Other options are $y = 0$ (the agent buys everything at the beginning) and $y = 1$ (one ticket per flight is not bought until after 4:00).

[8] This means that 8 t-tests will be computed if we have 2 instances of type A and 4 of type B etc. We consider the difference between the scores of A and B to be significant, if almost all the tests produce values below 10%.

ments and non-aggressive agents (WB-M2L) do relatively better in more competitive environments, but still not good enough compared to WB-M2M and WB-M2H agents. Overall WB-M2M (medium aggressiveness) performs comparably or better than the other agents in every instance (figure 2). Agents WB-M2L are at a disadvantage compared to the other agents because they do not bid aggressively enough to acquire the hotel rooms that they need. When an agent fails to get a hotel room it needs, its score suffers a double penalty: (i) it will have to buy at least one more plane ticket at a high price in order to complete the itinerary, or else it will end up wasting at least some of the other commodities it has already bought for that itinerary and (ii) since the arrival and/or departure date will probably be further away from the customer's preference and the stay will be shorter (hence less entertainment tickets can be assigned), there is a significant utility penalty for the new itinerary. On the other hand, aggressive agents (WB-M2H) will not face this problem and they will score well in the case that prices do not go up. In the case that there are a lot of them in the game though, the price wars will hurt them more than other agents. The reasons for this are: (i) aggressive agents will pay more than other agents, since the prices will rise faster for the rooms that they need the most in comparison to other rooms, which are needed mostly by less aggressive agents, and (ii) the utility penalty for losing a hotel room becomes comparable to the price paid for buying the room, so non-aggressive agents suffer only a relatively small penalty for being overbid. Agent WB-M2M performs well in every situation, since it bids enough to maximize the probability that it is not outbid for critical rooms, and avoids "price wars" to a larger degree then WB-M2H.

The last experiments intended to further explore the trade-off of bidding early for plane tickets against waiting more in order to gain more flexibility in planning. Initially we run an experiment with 2 instances of each of the following agents: WB-M2M and WB-M2H (since they performed best in the previous experiments) together with WB-M1M (an agent that bids for most of its tickets at the beginning) and WB-M0H (this agent bids at the beginning for all the plane tickets it needs and bids aggressively for hotel rooms.[9] We ran 78 games and observed that WB-M2M scores slightly higher than the other agents, while WB-M2H scores slightly lower. These results are however not statistically significant. For the next experiment we ran 2 instances of WB-M2M against 6 of WB-M0H to see how the latter (early-bidding agent) performs in the system when there is a lot of competition. We observed that instances of WB-M2M try to stay clear of rooms whose price increases too much (usually, but not always, successfully), while the early-bidders do not have this choice due to the reduced flexibility in changing their plans. The WB-M2M's average a score of 1586 against 1224, the average score of the WB-M0H's; we ran 69 games and the differences in the scores were statistically significant. We then changed the roles and ran 6 WB-M2M's against 2 WB-M0H's, to see how the latter performed in a setting where they were the minority. We ran 343 games and the score differences were statistically not significant. We observed that the WB-M0H's scored on average close to the score of the WB-M2M's and that in comparison to the previous experiment the scores for the WB-M2M's had decreased a little bit (1540) while the scores for the WB-M0H's had increased significantly (1517). These results allow us to conclude that it is usually beneficial not to bid for everything at the beginning of the game.

We are continuing our last set of experiments in order to increase the statistical confidence in the interpretation of the results so far. This is quite a time-consuming process, since each game is run at 20 minute intervals[10]. It took close to 2000 runs (about 4 weeks of continuous running time) to get the controlled experiment results and some 1000 more for our observations during the competition.

Results of the Trading Agent Competition

The semifinals and finals of the 2^{nd} International TAC were held on October 12, 2001 during the 3^{rd} ACM Conference on e-Commerce held in Tampa. The preliminary and seeding rounds were held the month before, so that teams would have the opportunity to improve their strategies over time. Out of 27 teams (belonging to 19 different institutions), 16 teams were invited to participate in the semi-finals and the best 8 advanced to the finals. The 4 top scoring agents (scores in parentheses) were: livingagents (3670), ATTac (3622), WhiteBear (3513) and Urlaub01 (3421). The *White Bear* variant we used in the competition was **WB-M2M**.

The scores in the finals were higher than in the previous rounds, because most teams had learned (as we also did) that it was generally better to have a more adaptive agent than to bid too aggressively.[11] A surprising exception to this rule was livingagents which followed a strategy similar to WB-M0H with the addition that it used historical prices to approximate closing prices. This agent capitalized on the fact that the other agents were careful not to be very aggressive and that prices remain quite low. Despite bidding aggressively for rooms, since prices did not go up, it was not penalized for this behavior. The plans that it had formed at the beginning of the game were thus easy to implement, since all it had to do was to make sure that it got all the rooms it needed, which it accomplished by bidding aggressively. In an environment where at least some of the other agents would bid more aggressively, this agent would probably be penalized quite severely for its strategy. We observed this behavior in the controlled experiments that we ran. ATTac went much further in its learning effort: it learned a model for the prices of the entertainment tickets and the hotel rooms based on parameters like the participating agents and the closing order for hotel auctions in previous games. Of course this makes the agent design more tailored to the

[9] An early-bidder must be aggressive, because if it fails to get a room, it will pay a substantial cost for changing its plan, due to the lack of flexibility in planning.

[10] This is a restriction of the game and the TAC server

[11] This was demonstrated by the second controlled experiment that we ran as well.

particular environment of the competition,[12] but it does improve the performance of the agent.

Conclusions

In this paper we have proposed an architecture for bidding strategically in simultaneous auctions. We have demonstrated how to maximize the flexibility (actions available, information etc.) of the agent while minimizing the cost that it has to pay for this benefit, and that by using simple knowledge (like modeling prices) of the domain it can make this choice more intelligently and improve its performance even more. We also showed that bidding aggressively is not a panacea and established that an agent, who is just aggressive enough to implement its plan efficiently, outperforms overall agents who are either not aggressive enough or who are too aggressive. Finally we established that even though generating a good plan is crucial for the agent to maximize its utility, the greedy algorithm that we used was more than capable to help the agent produce comparable results with other agents that use a slower provably optimal algorithm. One of the primary benefits of our architecture is that it is able to combine seamlessly both principled methods and methods based on empirical knowledge, which is the reason why it performed so consistently well in the TAC. Overall our agent is adaptive, versatile, fast and robust and its elements are general enough to work well under any situation that requires bidding in multiple simultaneous auctions.

In the future we will continue to run experiments in order to further determine parameters that affect the performance of agents in multi-agent systems. We also intend to incorporate learning into our agent to evaluate how much this improves performance.

Acknowledgements We would like to thank the TAC team for their technical support during the running of our experiments and the competition (over 3000 games). This research was supported in part under grants IIS-9734128 (career award) and BR-3830 (Sloan).

References

Anthony, P.; Hall, W.; Dang, V.; and Jennings, N. R. 2001. Autonomous agents for participating in multiple on-line auctions. In *Proc IJCAI Workshop on E-Business and Intelligent Web*, 54–64.

Fujishima, Y.; Leyton-Brown, K.; and Shoham, Y. 1999. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proc of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, 548–553.

Greenwald, A. R., and Boyan, J. 2001. Bid determination for simultaneous auctions. In *Proc of the 3rd ACM Conference on Electronic Commerce (EC-01)*, 115–124 and 210–212.

Greenwald, A., and Stone, P. 2001. Autonomous bidding agents in the trading agent competition. *IEEE Internet Computing, April*.

Greenwald, A. R.; Kephart, J. O.; and Tesauro, G. J. 1999. Strategic pricebot dynamics. In *Proceedings of the ACM Conference on Electronic Commerce (EC-99)*, 58–67.

Klemperer, P. 1999. Auction theory: A guide to the literature. *Journal of Economic Surveys Vol13(3)*.

Preist, C.; Bartolini, C.; and Phillips, I. 2001. Algorithm design for agents which participate in multiple simultaneous auctions. *In Agent Mediated Electronic Commerce III (LNAI), Springer-Verlag, Berlin* 139–154.

Sandholm, T., and Suri, S. 2000. Improved algorithms for optimal winner determination in combinatorial auctions. In *Proc 7th Conference on Artificial Intelligence (AAAI-00)*, 90–97.

Stone, P.; Littman, M. L.; Singh, S.; and Kearns, M. 2001. ATTac-2000: an adaptive autonomous bidding agent. In *Proc 5th International Conference on Autonomous Agents*, 238–245.

Wellman, M. P.; Wurman, P. R.; O'Malley, K.; Bangera, R.; de Lin, S.; Reeves, D.; and Walsh, W. E. 2001. Designing the market game for tac. *IEEE Internet Computing, April*.

[12] If the participants or their strategies change drastically, the price prediction would most likely become significantly less accurate.