SouthamptonTAC: An Adaptive Autonomous Trading Agent

MINGHUA HE and NICHOLAS R. JENNINGS
University of Southampton

Software agents are increasingly being used to represent humans in on-line auctions. Such agents have the advantages of being able to systematically monitor a wide variety of auctions and then make rapid decisions about what bids to place in what auctions. They can do this continuously and repetitively without losing concentration. Moreover, in complex multiple auction settings, agents may need to modify their behavior in one auction depending on what is happening in another. To provide a means of evaluating and comparing (benchmarking) research methods in this area, the Trading Agent Competition (TAC) was established. This competition involves a number of agents bidding against one another in a number of related auctions (operating different protocols) to purchase travel packages for customers. Against this background, this artcle describes the design, implementation and evaluation of our adaptive autonomous trading agent, SouthamptonTAC, one of the most successful participants in TAC 2002.

Categories and Subject Descriptors: I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*intelligent agents*; K.4.4 [**Computers and Society**]: Electronic Commerce

General Terms: Algorithms, Design, Experimentation

Additional Key Words and Phrases: On-line auctions, bidding strategy, trading agent competition

1. INTRODUCTION

On-line auctions are a key component of many e-commerce systems [He et al. 2003] and software agents are increasingly being used in such settings in order to make effective and efficient purchases. Given the potential and the importance of using agents in on-line auction settings, there has been considered a research endeavor in developing bidding strategies for multiple auctions [Anthony and Jennings 2003] (see Section 4 for more details). Given this fact, it was decided to establish an International Trading Agent Competition (TAC) as a means of comparing and evaluating work in this area [Wellman et al. 2001]. The TAC has been set up so that there is no optimal bidding strategy that is guaranteed to always win. This is because an agent's decision making involves

Authors' address: Department of Electronics and Computer Science, University of Southampton, Highfield, Southampton, SO17 1BJ, United Kingdom; email: {mh00r,nrj}@ecs.soton.ac.uk.

Table I. SouthamptonTAC's Customer Preferences for Game 4594. PAD and PDD Stand for Preferred Arrival and Preferred Departure Date. HV Stands for the Reservation Value of Staying in the Tampa Tower Hotel, and WV, PV and MV Stand for the Utility Associated with Attending Alligator Wrestling, the Amusement Park and the Museum

Customer	PAD	PDD	HV	WV	PV	MV
1	Day 3	Day 5	142	40	109	113
2	Day 1	Day 2	80	21	108	8
3	Day 4	Day 5	79	149	32	195
4	Day 1	Day 4	131	81	83	179
5	Day 2	Day 4	134	50	155	115
6	Day 2	Day 4	135	75	127	48
7	Day 3	Day 4	98	199	20	39
8	Day 1	Day 4	120	69	123	104

high degrees of uncertainty caused by the random features of the game, the opponents' strategies and the particular combination of opponents. Against this background, this article reports upon the design, implementation and evaluation of our particular trading agent (called *SouthamptonTAC*) in the 2002 competition.[1]

In a TAC trading game, there are eight software agents (entrants to the competition) that compete against each other in a variety of auctions to assemble travel packages for their individual customers according to their preferences for the trip.[2] A valid travel package for an individual customer consists of (i) a round trip flight during a 5-day period (between TACtown and Tampa) and (ii) a stay at the same hotel[3] for every night between their arrival and departure dates. Moreover, arranging appropriate entertainment events during the trip increases the utility for the customers. The objective of each agent is to maximise the total satisfaction of its eight customers (i.e., the sum of the customers' utilities). Customers have individual preferences over which days they want to be in Tampa, the type of hotel they stay in, and which entertainment they want to attend. This data is randomly generated by the TAC server in each game (see Table I for an example).

Each agent communicates with the TAC server through a TCP-based agent programming interface in order to get current market information and to place its bids. An individual game lasts 12 minutes and involves 28 auctions. Each of the three good types are traded in an auction with different rules:

—*Flights.* TACAIR is the only airline selling flights (placing asks). Tickets for these flights are unlimited and are sold in *single seller auctions*. There are eight such auctions (TACtown to Tampa (day 1 to 4) and back (day 2 to 5)). Flight ask prices update randomly, every 24 to 32 seconds, by a value drawn from a range determined by the elapsed auction time and a randomly drawn value. Flight auctions clear continuously during the game. Thus, any

[1]More details of this competition and its 26 entrants can be found at http://www.sics.se/tac.

[2]These packages are assembled by the agent bidding in a number of auctions in which the other bidders are other competition entrants.

[3]Customers are not allowed to change their hotels during the stay.

buy bid an agent makes that is not less than the current ask price will match immediately at the ask price. Those bids not matching immediately remain in the auction as standing bids. In most cases, the earlier the flight is bought, the cheaper it is.

—*Hotels.* There are two hotels: Tampa Towers (T) and Shoreline Shanties (S). T is nicer than S. Hotel rooms are traded in 16th price multi-unit English auctions. Overall, there are eight hotel auctions (for each combination of hotel and night apart from the last one), that close randomly one by one at the end of every minute after the 4th. A hotel auction clears and matches bids when it closes (i.e., 16 rooms are sold at the 16th highest price). While a given auction is open, its ask price is the current 16th highest price and this price is updated immediately in response to new bids. The price of other bids, such as the highest bid, is not known by the agents. No withdrawal of hotel bids is allowed. Suppose the current ask price is a, when an agent submits a new bid, two conditions must be satisfied for it to be accepted: (i) it must offer to buy at least one unit at a price of $a + 1$ or greater; (ii) if the agent's current bid would have resulted in the purchase of q units in the current state, the new bid must offer to buy at least q units at $a + 1$ or greater.

—*Entertainment.* Each agent is randomly endowed with 12 entertainment tickets at the beginning of the game. All agents can trade their tickets in a continuous double auction (CDA). Overall, there are 12 CDAs (for each kind of entertainment for each of days 1 to 4). Bids match at the price of the standing bid in the CDA. An entertainment package is feasible if none of the tickets are for events on the same day and all the tickets coincide with the nights the customer is in town. No additional utility is obtained for a customer attending the same type of entertainment more than once during the trip.

A customer's utility from a valid travel and entertainment package[4] is given by:

$$Utility = 1000 - TravelPenalty + HotelBonus + FunBonus,$$

where $TravelPenalty = 100 * (|AD - PAD| + |DD - PDD|)$ (AD and DD are the customer's actual arrival and departure dates), $HotelBonus$ is the bonus if the customer stays in T, and $FunBonus$ is the sum of the reservation values of all the entertainment a customer receives. To illustrate this, the allocations and scores for SouthamptonTAC, given the preferences in Table I, are shown in Table II. For example, the utility of customer 6 is obtained by the following:

$$TravelPenalty = 100 * (|AD - PAD| + |DD - PDD|) = 0,$$
$$HotelBonus = 135, \quad FunBonus = 75 + 127 + 0 = 202,$$
$$Utility = 1000 - 0 + 135 + 202 = 1337.$$

At the end of each game, the TAC scorer (on the TAC server) allocates the agent's travel goods to its individual customers optimally. The value for a particular allocation is the sum of the individual customer utilities (e.g. 9602). The

[4]An invalid travel package receives zero utility.

Table II. SouthamptonTAC's Customer Allocation from Game 4594. W, P, M Stand for Alligator Wrestling, Amusement Park and Museum and the Following Number Indicates the Date of the Entertainment

Customer	AD	DD	Hotel	Entertainment	Utility
1	Day 3	Day 5	T	P3,M4	1364
2	Day 1	Day 2	T	P1	1188
3	Day 4	Day 5	T	M4	1274
4	Day 1	Day 2	T	M1	1110
5	Day 3	Day 4	T	P3	1189
6	Day 2	Day 4	T	W2, P3	1337
7	Day 2	Day 3	T	W2	1097
8	Day 1	Day 2	T	P1	1043
				Total utility:	9602

agent's final score is then the value of this allocation minus the cost of procuring the goods.

Designing a bidding strategy for the TAC auction context is a challenging problem. First, there are interdependencies. These exist between different kinds of auctions (e.g., flights will be useless if the hotel rooms are not available); between different dates within the same kind of auction (e.g., customers must stay in the same hotel during their stay); and between same day, same kind counterpart auctions[5] (e.g., if the price of T1 is high, the customer can change to S1). Second, the bidding involves uncertainty. For example, flight prices start randomly and change continuously in a random fashion; one randomly selected hotel auction closes from the 4th to 11th minute. Third, trade-offs exist in bidding. For example, in flight auctions, if an agent buys all the flight tickets very early, it may fail to buy the necessary hotel rooms that the flights require.

The remainder of the article is organized as follows: Section 2 presents the design of our agent. Section 3 describes the result of TAC-02 and evaluates the performance of our agent in different environments. Section 4 discusses the work of other teams participating in the TAC and Section 5 concludes.

2. SOUTHAMPTONTAC

SouthamptonTAC is an adaptive agent that varies its bidding strategy according to its perception of the prevailing market conditions.

2.1 Classifying TAC Environments

Our post hoc analysis of the TAC-01 competition [He and Jennings 2002] shows that an agent's performance depends heavily on the risk attitude of its opponents. Here a *risk-averse* agent is one that buys a small number of flight tickets at the beginning of the game and that bids for hotels according to the situation as the game progresses. This kind of agent is highly flexible and copes well when there is a significant degree of competition and the hotel prices are high

[5]For the auction of the same day, T and S are called their counterpart auctions. For example, the counterpart auction of T1 is S1 and the counterpart of S1 is T1. Also, we will use the abbreviation Tn and Sn ($1 \leq n \leq 4$) for staying in the indicated hotel on a particular day.

(see below). In contrast, a *risk-seeking* agent buys a large number of flight tickets at the beginning of the game and seldomly changes the travel plan of its customers during the game. This kind of agent does well in environments in which hotels are cheap. For example, when a hotel price goes up sharply, a risk-averse agent would stop bidding on that hotel (changing the stay to a counterpart hotel or reducing the trip period). In contrast, a risk-seeking agent will insist on bidding for that hotel, although the price is very high. In so doing, it hopes that the price will eventually stabilize (hence the risk). The consequence of this variety is that, for broadly the same situation, different agents can bring about widely varying final prices. Based on the analysis reported in He and Jennings [2002], we identify the following types of TAC environment:

—*Competitive environments* where the prices of the hotels are (very) high. This is caused by (a) the high bid prices that agents place; (b) the fact that some agents insist on bidding for hotels even when their ask price becomes high; and (c) the fact that some agents increase their bids sharply rather than gradually. For example, in game 4594, the prices of T (S) are (in the increasing order of day): 5 (6), 238 (557), 155 (102), and 40 (11). For most customers in this game, it is beneficial for an agent to reduce the stay to a single day (either day 1 or day 4). To achieve this, however, the agent needs to be flexible. Specifically, it cannot buy all the flights at the very beginning of the game; otherwise, when the hotel prices rise to high values, it has to give up the travel package for some customers or pay these high prices for hotels. Being predictive is also important. By predicting the price of the hotels, the agent can make alternative plans to cope with the very high prices.

—*Noncompetitive environments* where there is very little competition for hotels and an agent can obtain the rooms it wants at low prices. For example, in game 6341, there is very little competition and the closing prices for T (S) are 7 (12), 92 (27), 70 (53) and 62 (7). In this situation, the best strategy is to buy all flights earlier; since the agents can always get the hotels they want.

—*Semicompetitive environments* where prices are medium. There is competition, but it is not very severe. For example, in game 444, the clearing prices for T (S) are 5 (2), 128 (71), 128 (60) and 116 (3).

2.2 The Agent Architecture

Figure 1 overviews the SouthamptonTAC agent. It connects to the TAC server in a continuous series of rounds (lasting between 2 and 4 seconds). In each round, our agent processes this ask/bid information in "Bids preprocessor" to get the flight prices, goods it actually owns and may possibly own (hotel rooms) and its currently active bids. "Hotel price predictor" is used to predict the likely clearing price of each hotel auction (see Section 2.3). All of this information is then input to "Allocator," which calculates the optimal distribution of goods to customers given the current situation (using linear and integer programming techniques). Given this assignment, the agent then determines its subsequent bidding actions. "Hotel bid adjustor" takes the allocator's output, the agent's current active bids, the hotel auction's ask prices, as well as the predicted prices

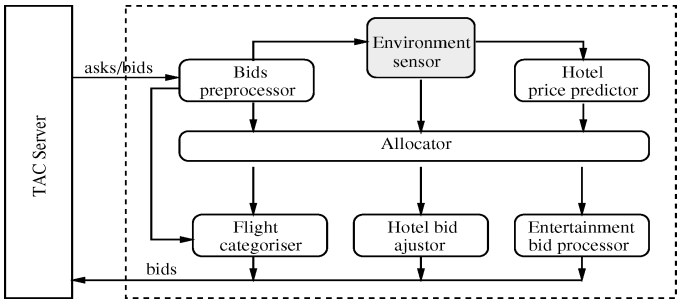

Fig. 1. Overview of the SouthamptonTAC-02 agent.

and decides whether to bid. "Flight categorizer" classifies each flight auction in accordance with its expected change of price and, based on this, decides when to bid and how many trips to bid for. "Entertainment bid processor" determines the type and the amount of entertainment to bid for. All of this is broadly the same as the SouthamptonTAC-01 agent, and more details can be found in He and Jennings [2002].

Where SouthamptonTAC-02 differs from its predecessor is in the way that it performs hotel price prediction (Section 2.3) and in having an environment sensor in the architecture. This component (described in more detail in Section 2.4) aims to determine what type of environment the agent is presently situated in (as detailed in Section 2.1). The reason for doing this is so that the agent can adapt its bidding strategy accordingly (see Section 2.5).

2.3 Hotel Price Prediction

Like its predecessor, SouthamptonTAC uses fuzzy reasoning techniques (specifically the Sugeno controller [Sugeno 1985]) to predict hotel clearing prices (see Section 3.3 for an evaluation of the effectiveness of our approach). SouthamptonTAC-01 used two rule bases to make its predictions: (i) for when both the good and bad hotels are open and (ii) for when the counterpart auction was closed. However, we found that our predictions in the latter case could be improved if we separated out the cases in which the counterpart auction had just closed (within the last minute) and when it had been closed for a longer period of time. This difference occurs because hotel prices change more rapidly and with a different pattern when the counterpart has just closed. When the counterpart auction has been closed for a longer period, the changes are smaller. We also simplified the prediction rules for the case where both auctions are still open. For example, in some cases, the current price in the counterpart auction is not considered because we found the price change to be a better indicator than the current price. In more detail, the three rule bases for SouthamptonTAC-02 are given in Tables III to V. The factors considered in the predictions are the price of the hotel (P), the price of the counterpart hotel (CP), the price change in the previous minute (C) and the previous price change of the counterpart hotel (when it closed) (CC). In these tables, the hotel ask prices (P and CP) are expressed in the fuzzy linguistic terms: *high*, *medium*, and *low* and the price changes (C and CC) in the fuzzy linguistic terms *quick*, *medium*, and *slow*.

Table III. Fuzzy Rule Base When Counterpart Auction is Open

IF P is *high* and C is *quick* THEN Δ is *big*.
IF P is *high* and CP is *high* and C is *not-quick* THEN Δ is *big*.
IF P is *high* and CP is *not-high* and C is *not-quick* THEN Δ is *medium*.
IF P is *low* and CP is *high* THEN Δ is *medium*.
IF P is *low* and CP is *not-high* THEN Δ is *small*.
IF P is *medium* and CP is *high* THEN Δ is *medium*.
IF P is *medium* and CP is *not-high* and C is *not-slow* THEN Δ is *medium*.
IF P is *medium* and CP is *not-high* and C is *slow* THEN Δ is *small*.

Table IV. Fuzzy Rule Base When Counterpart Auction Just Closed

IF P is *high* and C is *not-slow* and CC is *quick* THEN Δ is *very-big*.
IF P is *high* and C is *not-slow* and CC is *not-quick* THEN Δ is *big*.
IF P is *high* and C is *slow* and CC is *quick* THEN Δ is *big*.
IF P is *high* and C is *slow* and CC is *not-quick* THEN Δ is *medium*.
IF P is *medium* and C is *quick* THEN Δ is *big*.
IF P is *medium* and C is *medium* and CC is *quick* THEN Δ is *big*.
IF P is *medium* and C is *medium* and CC is *not-quick* THEN Δ is *medium*.
IF P is *medium* and C is *slow* and CC is *quick* THEN Δ is *medium*.
IF P is *medium* and C is *slow* and CC is *not-quick* THEN Δ is *small*.
IF P is *low* and C is *slow* and CC is *quick* THEN Δ is *medium*.
IF P is *low* and C is *not-slow* THEN Δ is *medium*.
IF P is *low* and C is *slow* and CC is *not-quick* THEN Δ is *small*.

Table V. Fuzzy Rule Base When Counterpart Auction has Closed for More than One Minute

IF P is *high* and C is *not-slow* THEN Δ is *big*.
IF P is *high* and C is *slow* THEN Δ is *medium*.
IF P is *medium* and C is *quick* THEN Δ is *big*.
IF P is *medium* and C is *medium* THEN Δ is *medium*.
IF P is *not-high* and C is *slow* THEN Δ is *small*.
IF P is *low* and C is *not-slow* THEN Δ is *medium*.

$\Delta \in$ {small, medium, big, very-big} is the increase that is added to the current price to obtain the predicted clearing price.

2.4 TAC Environment Recognition

We treat the environment recognition problem as one of fuzzy pattern recognition since it is impossible to precisely determine the type while the game is running. To this end, we therefore apply the maximum similarity principle [Pedrycz 1990] in the environment sensor component of the agent architecture. This recognition process is used in two cases: before a game starts and during a game. Before a game starts, the agent calculates the average hotel closing prices of the previous 10 games[6] from the price history and uses the maximum average price as a reference price to classify the environment in a given game. We use the maximum price in this fashion since if one price is high it is likely

[6]We chose ten (based on experience of playing the game) as a suitable indicator that is sufficiently stable to not be influenced by atypical game outcomes, but sufficiently adaptive to respond to genuine changes in the patterns of games.

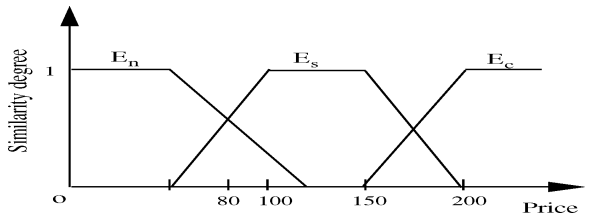

Fig. 2. Fuzzy sets of the three environment types.

that others will also be high and so the environment is competitive (*mutatis mutandis* when the reference price is low or medium). During a game, the agent continuously monitors the current hotel prices and records the current maximum price to see if the environment type changes from its initial prediction. More formally, let T_i (S_i) represent T (S) on day i, and P_{T_i} (P_{S_i}) represent the current price if the agent is monitoring a running auction or the average history price if it is making its initial assessment of the environment of T_i (S_i). Suppose the prices of $T_1, \ldots, T_4, S_1, \ldots, S_4$, are $P_{T_1}, \ldots, P_{T_4}, P_{S_1}, \ldots, P_{S_4}$. Then, $P_{\max}$ (the maximum hotel price) is simply:

$$P_{\max} = \max(P_{T_1}, P_{T_2}, P_{T_3}, P_{T_4}, P_{S_1}, P_{S_2}, P_{S_3}, P_{S_4}).$$

Let E_c, E_s, and E_n correspond to the fuzzy sets that represent competitive, semi-competitive and non-competitive environments respectively (see Figure 2). Now the type of environment (ε) can be determined by ascertaining which of the fuzzy sets the reference price has the strongest membership to. Thus, if:

$$E_x(P_{\max}) = \text{argmax}\{E_c(P_{\max}), E_s(P_{\max}), E_n(P_{\max})\},$$

then ε is of environment type E_x, where $x \in \{c, s, n\}$ and $E_c(x)$, $E_s(x)$, and $E_n(x)$ are the similarity functions for the fuzzy sets E_c, E_s, and E_n. These similarity functions (denoted μ) capture how much the hotel price belongs to each of the different environments and they are defined as follows:

$$\mu_{E_N}(x) = \begin{cases} 1 & x < 50, \\ 0 & x > 120, \\ \frac{120-x}{70} & 50 \le x \le 120. \end{cases}$$

$$\mu_{E_S}(x) = \begin{cases} 1 & 100 < x < 150, \\ 0 & x > 200 \text{ or } x < 50, \\ \frac{x-50}{50} & 50 \le x \le 100, \\ \frac{200-x}{50} & 150 \le x \le 200. \end{cases}$$

$$\mu_{E_C}(x) = \begin{cases} 1 & x > 200, \\ 0 & x < 150, \\ \frac{x-150}{50} & 150 \le x \le 200. \end{cases}$$

2.5 Varying the Bidding Strategy

After our experiences in TAC-01, we came to believe that there is no single best strategy that can deal with all the different types of TAC environment. For example, a risk-seeking agent that always allocates the optimal travel package for its customers and buys flights earlier is highly effective in noncompetitive environments. This is because there is little competition in hotel bidding and the agent can always obtain what it wants. On the other hand, delaying buying flights and shortening the stay of customers works well in competitive games. For this reason, SouthamptonTAC dynamically varies its bidding strategy according to its assessment of the environment type (see Section 3.4 for an evaluation of the effectiveness of being able to do this). In games it deems noncompetitive, SouthamptonTAC buys all of its flight tickets at the beginning of the game and never changes the travel plan of its clients (unless it senses a change in the environment). In this way, it avoids buying extra hotels which cost extra money. Also, the agent can receive optimal utility by not shortening the stay of its customers. In competitive games, our agent buys flights according to its assessment of the flight category.[7] In these games, the agent may alter its customers' travel plans in order to avoid staying in expensive hotels for long periods. In semicompetitive games, the agent behaves in between these two strategies; it buys most of the flights earlier and will only change travel plans if a significant improvement can be obtained.

3. EVALUATION

Our evaluation of SouthamptonTAC is composed of two components: (i) the results from the 2002 competition itself and (ii) our post-hoc systematic analysis in a range of controlled environments.

3.1 TAC-02 Results

TAC-02 consisted of a preliminary round (mainly used for practice and fine tuning), a seeding round, the semifinals and the final round. The seeding round determined groupings for the semifinals and involved about 440 games. To this end, Table VI shows the result of each agent's relative score to SouthamptonTAC. Note there is less than 2 points difference between ATTac and SouthamptonTAC and given the random features of the game their performance should be considered as broadly similar. The top 16 agents were organized into two "heats" for the semifinals based on their position in the seeding round and the first four teams in both heats entered into the final round. Table VII shows the scores (again relative to our agent) of all the agents in this final round. Again the difference between SouthamptonTAC and the top agent is small, less than 0.8% than whitebear.

Overall, in the course of the competition some 606 games were played and SouthamptonTAC had a higher mean score than both ATTac and whitebear.

[7]The agent continuously monitors the flight price changes and categorises each flight according to its rate of change. Based on the flight category, it decides when to buy the flights in a flexible way (e.g., rapidly rising flights are bought quickly, whereas slow rising ones are bought near the end).

Table VI. Result of Seeding Round (440 Games)

Rank	Agent	avg(-10 worst)	Avg
1	ATTac	3129.5+1.8	3033.5+4.2
2	SouthamptonTAC	3129.5	3033.5
3	UMBCTAC	3129.5−11.1	3033.5−16.6
4	livingagents	3129.5−38.1	3033.5−24.9
5	cuhk	3129.5−74	3033.5−62.1
6	Thalis	3129.5−129.8	3033.5−131.9
7	whitebear	3129.5−163.9	3033.5−158.2
8	RoxyBot	3129.5−274.2	3033.5−300.8

Table VII. Result of Final Round (32 Games)

Rank	Agent	Avg(-worst)	Avg
1	whitebear	3492+64.4	3385.5+27.3
2	SouthamptonTAC	3492	3385.5
3	Thalis	3492−140.8	3385.5−139.2
4	UMBCTAC	3492−171.4	3385.5−149.9
5	Walverine	3492−176.4	3385.5−175.9
6	livingagents	3492−182.2	3385.5−204.6
7	kavayaH	3492−242.2	3385.5−286.0
8	cuhk	3492−244.2	3385.5−316.7

We believe that this large number of games and the very nature of the competition mean that the difference in the trader's scores reflect true differences in the performance of the agents' strategies. Thus, we believe SouthamptonTAC performs successfully in a wide range of situations.

3.2 Controlled Experiments

To evaluate the performance of our agent in a more systematic fashion than is possible in the competition, we decided to run a series of controlled experiments. To do this, we devised two competitor agents that adopt strategies consistent with the broad classes of behavior that were observed in the competition:

—*Risk-seeking agent (RS-agent)*. This is based on the behavior of livingagents, UMBCTAC, and Walverine agents (see Section 4 for more details). This agent buys all the flights tickets at the beginning of the game, bids aggressively in hotel auctions and never changes the plans for its customers.

—*Risk-averse agent (RA-agent)*. This is based on the behavior of SouthamptonTAC-01, Retsina, and sics agent (see Section 4 for more details). This agent buys a small number of flight tickets at the beginning of the game to leave some flexibility and it will change the customers' travel plans according to how the game unfolds.

For both these types of agents, as well as for SouthamptonTAC, a record is kept of the closing price history and the initial travel plans for the customers are calculated based on the average price of this history.

The set-up of the experiment is shown in Table VIII where it can be seen that there are 36 different cases which cover all possible combinations of SouthamptonTAC, RA-agents and RS-agents given that there can only be eight agents in

Table VIII. Experiment Set-Up for Controlled Experiments. The Light Grey Area Indicates Competitive Environments and Dark Grey Non-Competitive Ones

Number of RS-agents	Number of RA-agents: 0	1	2	3	4	5	6	7
0	8	7	6	5	4	3	2	1
1	7	6	5	4	3	2	1	
2	6	5	4	3	2	1		
3	5	4	3	2	1			
4	4	3	2	1				
5	3	2	1					
6	2	1						
7	1							

↑ Number of ← SouthamptonTAC agents

one game. For example, in the case where the number of RA-agents is 2 and the number of RS-agents is 1, there will be 5 SouthamptonTAC agents. For each case, between 50–100[8] games were played to test the performance of each kind of agent. In this way, it is possible to produce a wide range of environments, from competitive to noncompetitive, and to evaluate the corresponding performance of the different types of agents and the broad behavior trends. The following Conjectures were used as an approximate guide for designing the experiments.

Conjecture 1. The more RS-agents there are in the game, the more competitive (see Section 2.1) it will be and the more RA-agents there are, the less competitive it will be (shown by the shading in Table VIII).

Conjecture 2. RS-agents will do well in noncompetitive environments and RA-agents will do well in competitive ones.

In terms of Conjecture 1, the average hotel clearing price for those environment marked as competitive (in Table VIII) is 240 and for those environment marked as noncompetitive it is 67. Thus, Conjecture 1 can be seen to hold.

We now start to analyze the performance in the different environments. Figure 3 shows the performance surface of SouthamptonTAC in the different cases. The x-axis and y-axis represent the number of RS-agents (N_{RS}) and RA-agents (N_{RA}), thus, the number of SouthamptonTAC is $8-N_{RS}-N_{RA}$. The z-axis shows the average score[9] of SouthamptonTAC. The higher the score, the better the agent performs. Figures 4 and 5 show the performance of the RS-agents and RA-agents on similar graphs.

As shown in Figure 3, SouthamptonTAC does best, obtains the highest score, in competitive games (i.e., where the number of RS-agents is big). This is due to the adaptive nature of its strategy. When it finds the game competitive, it alters its strategy in the direction of being risk-averse. In noncompetitive environments, where there are many RA-agents, SouthamptonTAC also does well since it adapts its strategy to bid aggressively because it can always obtain the

[8]This number differs from game to game. The experiment for a single case stops when the relative scores of the agents become stable.

[9]Suppose there are m SouthamptonTAC agents, and the average scores of these agents are $s_1, s_2, \ldots, s_m$. Then the average score shown on the z-axis is $(\sum_{i=1}^{m} s_i)/m$.

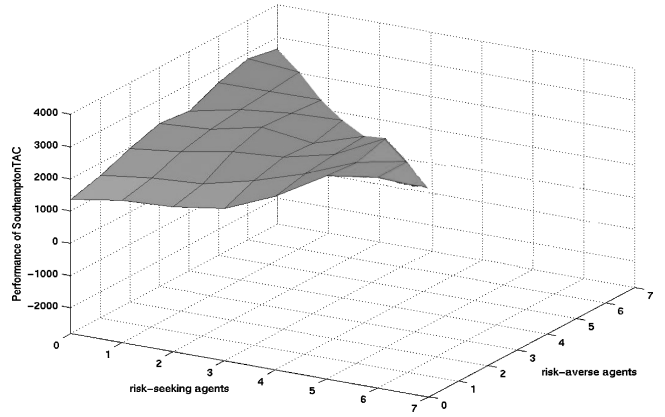

Fig. 3. Performance of SouthamptonTAC in different environments.

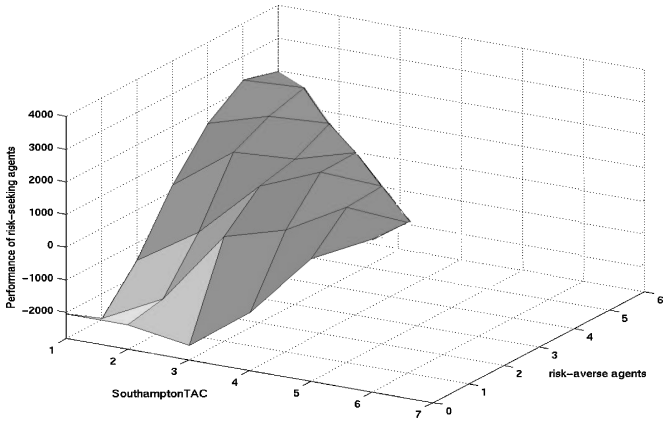

Fig. 4. Performance of Risk-seeking agents in different environments.

goods it wants. Both of these observations are consistent with Conjecture 2. The worst situation for SouthamptonTAC is when all the players are like itself. This is because the competitive tendency of the agents causes the hotel prices to rise to moderate levels and then many of the agents change their customers' travel plans at approximately the same time. This switching behaviour causes the counterpart hotel prices to rise (because of increased competition) and the agents to have unused flights or hotel rooms bought on account of their previous travel plans. For RS-agents, as shown in Figure 4, the results also support Conjecture 2. RS-agents behave very well in noncompetitive games and their performance decreases rapidly as the number of RS-agents increases. This happens because as more agents bid aggressively, the hotel closing prices get higher. RA-agents behave best in competitive environments when there are many RS-agents, perform adequately in noncompetitive games and worst in semicompetitive games when there are a few RS-agents and SouthamptonTAC agents (see Figure 5). In the latter two cases, RA-agents change their customers'

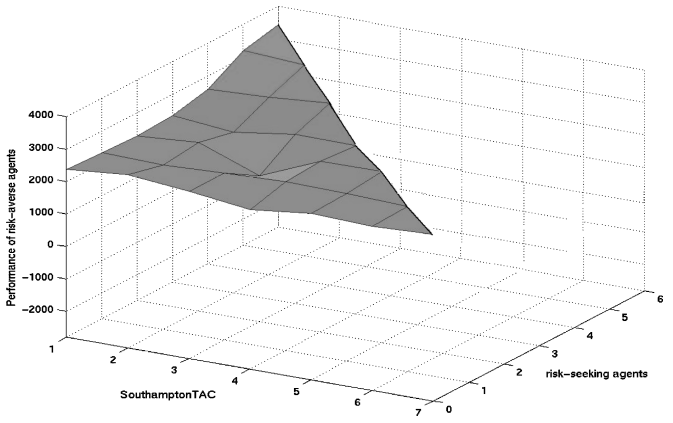

Fig. 5. Performance of Risk-averse agents in different environments.

travel packages reasonably often and this causes them to buy extra hotels and flights that they cannot subsequently use.

Moreover, from Figures 3 to 5, we find that the range of scores for each kind of agent are different; for SouthamptonTAC it is [1372, 3737], for RS-agents it is [−2742, 2374] and for RA-agents it is [1709, 3445]. Thus, the RA-agent has the narrowest score range and is the most stable agent. The RS-agent has the widest score range since its performance depends heavily on the environment it is situated in. SouthamptonTAC is in between, less stable than RA-agents (but able to obtain higher scores) but with a better worst performance than RS-agents.

While Figures 3 to 5 show the performance of a single type of agent in various environments, Figure 6 compares their scores. There are eight subfigures and each of them represents several cases of the above experiments.[10] We found that when the number of SouthamptonTAC agents is small (less than 4), they can always outperform both RS-agents and RA-agents (as shown in (e) to (h) and some cases in (a) to (d)). This is because SouthamptonTAC can successfully adapt itself in competitive games and become aggressive in noncompetitive ones. However, as we discussed previously, when the number of SouthamptonTAC agents is above 4, the agents exhibit similar behaviour and make the market less efficient. Generally, from (a) to (h), it can also be seen that profits for all agent types increase as the number of RA-agents increases (because these agents keep the hotel prices low).

3.3 Predicting Hotel Prices

Most of the agents in TAC engage in some form of hotel price prediction (see Section 4). Since, generally speaking, the more accurately the agent can predict these prices the more easily it can identify profitable actions. To this end, Table IX shows the accuracy of SouthamptonTAC's predictions on a minute by

[10]For example, in figure (c), there are two RS-agents, thus the horizontal axis represents the number of RA-agents and the vertical axis is the average score of the different agent types.

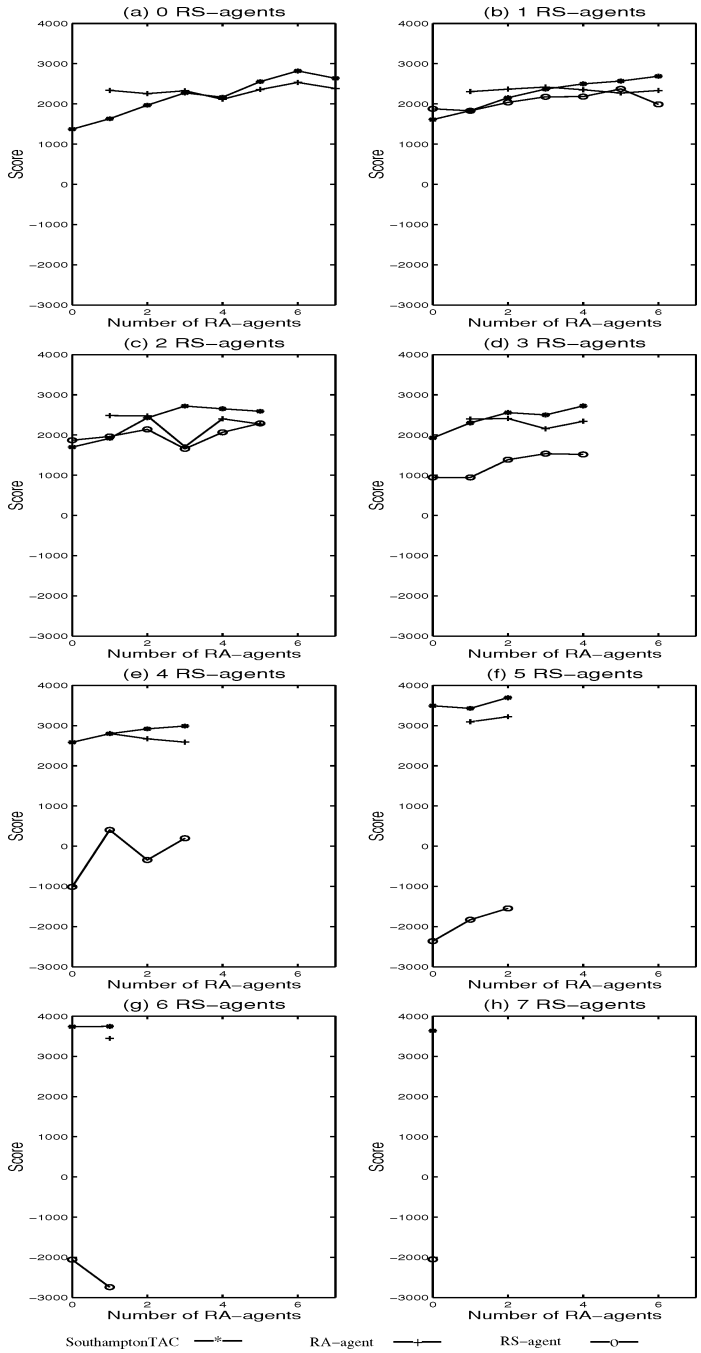

Fig. 6. Relative Performance of the agents in different environments.

Table IX. Actual vs. Predicted Hotel Prices. Positive Figure Means Over Prediction and Negative Figure Means Under Prediction

Hotel	4	5	6	7	8	9	10	11
T3	86							
T2	40	23						
S3	76	−62	50					
S4	39	9	9	9				
S2	83	11	−32	−53	−17			
S1	45	9	8	9	9	9		
T1	79	9	8	8	8	8	8	
T4	87	10	10	10	10	10	10	−17

Table X. Hotel Closing Price Prediction in Final Round

Closing order	Avg difference	Max difference	Min difference
1	64	174	8
2	30	103	2
3	29	144	8
4	38	149	1
5	32	115	8
6	30	111	3
7	27	87	8
8	20	62	9

minute basis for a single game (randomly chosen) in the final. The figures in the table are the difference between the predicted price and the actual price. Thus, a positive number means over prediction and a negative one means under prediction. As we can see, the trend is that the further into the game the predictions are made the more accurate they are. This is because at the beginning the agent can only work based on the price history of previous games. However, as the game progresses, more information is revealed (such as the closing order of the hotels, the current hotel prices and the relation between the hotels). This, in turn, means more accurate predictions can be made. This is important for our agent since it enables its flexible decision making to be based on more or less accurate information. In most cases, our agent tends to over predict the hotel closing prices. If the hotel prices are not very high, the agent will not suffer since it will not change the plan for its customers; whereas if the prices are very high, the agents may change the travel plans for its customers and therefore obtain a lower score (since it may have bought flights or rooms that it cannot now use). However, when hotel prices rise very quickly, our agent tends to under predict which can cause it to buy highly priced hotels (so reducing its profit).

Furthermore, Table X shows the difference between the predicted and actual hotel closing prices for the order in which they closed in the final. For example, for the hotel that closed first (whatever that happened to be in a particular game), the average difference is 64, the maximum difference is 174[11] and the

[11]This number is large and it occurred at the beginning of the final where the price history data was based upon the seeding round (which had very different outcomes from the final round).

Table XI. Comparison Among Agents (RS Means Risk Seeking, RA Risk Averse and RN Risk Neutral (Between RS and RA). —Means Information Unavailable)

Agent	Price Prediction	Allocator	Attitude
ATTac	machine learning	ILP	RN
cuhk	average prices	heuristic search	RN
livingagents	average prices	search	RS
PainInNEC	—	Genetic algorithm	RA
Retsina	price matrix	Markov chain	RA
RoxyBot	price distribution	heuristic search	RA
sics	price distribution	branch-and-bound search	RA
SouthamptonTAC	Fuzzy reasoning	ILP	RN
UMBCTAC	average price	heuristic search	RS
Walverine	Walrasian competitive equilibrium	ILP	RS
whitebear	average price	greedy search	RN

minimum is 8. These results are consistent with those of Table IX and show that the later a hotel closes, the more accurate our agent's prediction is.

3.4 Strategy Adaptation

To test the value of the agent being able to adapt its strategy during the course of a game, we compare the performance of our agent with a nonadaptive variant (called na-SouthamptonTAC) that is identical apart from the fact that it cannot change its strategy once a game has started. In each game, there was one SouthamptonTAC, one na-SouthamptonTAC and the remaining agents were drawn randomly from a pool of RS-agents and RA-agents. We ran this configuration for 164 games and computed the average score of each agent type. Our results were that the adaptive agent received an average score of 3138, the nonadaptive one an average of 2937 and the other agents an average of 1657 (RS-agents) and 2649 (RA-agents). This shows that being adaptive does indeed improve the agents' performance.

4. RELATED WORK

Related work on bidding in multiple heterogeneous auctions in general is discussed in the companion paper [Anthony and Jennings 2003]. So here we focus solely on examining agents developed for the TAC. Specially, we discuss the most successful agents from both TAC-01 and TAC-02 in Table XI. *ATTac* uses machine learning techniques to obtain a model of the price dynamics based on the past data (e.g., the data in the seeding round) to predict the closing prices of the hotels in the future. It also uses mixed-integer linear programming (ILP) to find the optimal allocation of the goods [Stone et al. 2001]. *cuhk* agent is composed of a cost estimator, an allocation and acquisition solver and bidders. It uses a greedy, heuristic search to find the travel packages for customers. *livingagents* [Fritschi and Dorer 2002] bases its decisions on closing price data for the various hotels in past games and it buys all the flights needed at the beginning of the game. It also buys/sells entertainment tickets at a fixed price of 80. It makes bids for the needed hotels only once during the game again at a

fixed price (of 1001). *PainInNEC*'s strategy is a combination of heuristics and a genetic algorithm based optimization method, which outputs the goods to buy and sell given the predicted auction clearing prices and customers' preferences. *Retsina* uses a Markov Chain Monte Carlo approach to allocate the goods to its customers and it uses a matrix learned from past games to predict the hotel's future prices. *RoxyBot* [Greenwald and Boyan 2001] decides the goods to bid for based on heuristic search techniques and applies a marginal utility calculator to determine the value of the goods. *sics* uses *pricelines* for price prediction and the optimiser performs branch-and-bound search for the best solutions. *UMBCTAC* balances the minimal risk and the maximum return to find the best travel plan for its customers. *Walverine* predicts the hotel closing prices by calculating the Walrasian competitive equilibrium of the game. *whitebear* uses a randomised greedy algorithm to calculate the price of each commodity bought or sold and uses Bayesian analysis to compute the minimum and average value of the flight's determinant factor.

As can be seen from the above discussion, the TAC agent designs incorporate a variety of AI techniques including fuzzy reasoning, machine learning, planning, Markov decision making and heuristic searching. Most agents keep a record of the hotel closing prices and use a variety of methods to predict subsequent hotel closing prices in order to allocate travel packages to customers. Moreover, a number of the agents adapt their bidding behavior in response to environmental changes. Such adaptation includes our agent varying its bidding behaviors (as described in Section 2.5); ATTac, which varies the number of flights it buys at the beginning of the game; and whitebear, which postpones some flight ticket purchases until after it learns the hotel prices.

5. CONCLUSIONS

SouthamptonTAC has been shown to be successful across a wide range of TAC environments. Naturally, the strategies that have been employed are tailored to the specific auction context of the competition (as is any agent strategy for any other auction context). Nevertheless, we believe that the TAC domain exhibits a number of characteristics that are common to many real-world, on-line trading environments. These attributes include a time constrained environment, network latency, unpredictable opponents, multiple heterogeneous auction types and the need to purchase interrelated goods. Given this, we believe that a number of insights and technologies from our work are applicable in a broader agent-mediated e-commerce context and our future work aims to exploit these. First, the uncertainty that is inherent in such situations means that a bidding agent needs to be able to adapt its behavior and strategy during the course of its interactions. While attempting to settle these things in advance and not responding to the prevailing context may sometimes work (even in repeated encounters), it can produce brittle behaviour that is not robust in a wide variety of circumstances. Nevertheless, some degree of prior analysis is essential to set the basic parameters to approximately correct values otherwise the agent may take a long time before it starts to perform effectively. Second, the fuzzy technologies we developed for making predictions, assessing the prevailing context

and adapting behaviour were shown to be computationally efficient and able to operate with relative simple information that is likely to be readily available in most trading contexts. Third, and most generally, effective and robust behavior requires a detailed understanding of the space of environmental possibilities and a careful evaluation of the broad responses that are desirable in such cases. Having obtained this, the specific behavior within such a framework can then be fine-tuned using appropriate analysis and adaptive technologies.

ACKNOWLEDGMENTS

During the project, we received help and support from many people. We would like to thank the TAC team at the University of Michigan (TAC-01) and Swedish Institute of Computer Science (TAC-02), including Michael Wellman, Kevin O'Malley, William E. Walsh, Daniel Reeves Chris Kiekintveld, Joakim Ericsson and Niclas Finne for setting up the TAC server and the competition and for responding so promptly to our questions and requests. We would also like to express our thanks to a number of people from The University of Southampton: Mike Luck, who went to the US to present our agent at TAC-01 at very short notice; Yan Zheng Wei, who helped us with agent connecting problems as well as monitoring the seeding round games in TAC-02; Alastair J. Riddoch, Jon Hallett and Tim Chown, who helped us with various network difficulties that we experienced in both competitions; and Xudong Luo for his encouragement and support during the whole course of the competition.

REFERENCES

ANTHONY, P. AND JENNINGS, N. 2003. Developing a bidding agent for multiple heterogeneous auctions. *ACM Trans. Internet Tech. 3*, 3 (Aug.), 185–217.

FRITSCHI, C. AND DORER, K. 2002. Agent-oriented software engineering for successful TAC participation. In *Proceedings 1st Joint Conference on Autonomous Agents and Multi-Agent Systems* (Bologna, Italy). 45–46.

GREENWALD, A. AND BOYAN, J. 2001. Bidding algorithms for simultaneous auctions: A case study. In *Proceedings of the 3rd ACM Conference on Electronic Commerce*. ACM, New York, 115–124.

HE, M. AND JENNINGS, N. R. 2002. SouthamptonTAC: Designing a successful trading agent. In *Proceedings of the 15th European Conference on Artificial Intelligence*, F. van Harmelen, Ed. IOS Press, Amsterdam, The Netherlands, 8–12.

HE, M., JENNINGS, N. R., AND LEUNG, H. F. 2003. On agent-mediated electronic commerce. *IEEE Trans. Knowl. Data Eng. 15*, 4 (July/Aug.).

PEDRYCZ, W. 1990. Fuzzy sets in pattern recognition methodology and methods. *Patt. Rec. 23*, 1, 121–146.

STONE, P., LITTMAN, M., SINGH, S., AND KEARNS, M. 2001. ATTac-2000: An adaptive autonomous bidding agent. *J. Artif. Int. Res. 15*, 189–206.

SUGENO, M. 1985. An introductory survey of fuzzy control. *Inf. Sci. 36*, 59–83.

WELLMAN, M., WURMAN, P., O'MALLEY, K., BANGERA, R., LIN, S., REEVES, D., AND WALSH, W. 2001. Designing the market game for a trading agent competition. *IEEE Internet Comput. 5*, 2, 43–51.

Received May 2002; revised November 2002 and January 2003; accepted January 2003