SouthamptonTAC: Designing a Successful Trading Agent

Minghua He and Nicholas R. Jennings[1]

Abstract. This paper describes the design, implementation and evaluation of SouthamptonTAC, an agent that participated in the Second International Trading Agent Competition. In the course of the competition's approximately 600 games, SouthamptonTAC achieved the highest mean score and the lowest standard deviation.

1 INTRODUCTION

The Second International Trading Agent Competition [3] (TAC) involved 27 agents developed by universities and research labs from around the world. In the course of the competition, some 600 games were played and our agent (SouthamptonTAC) registered the highest mean score and the lowest standard deviation. In a TAC trading game, there are 8 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. 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[2] 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 8 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 1 for an example).

Table 1. SouthamptonTAC's customer preferences for game 5722. 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	80	178	183	136
2	Day 3	Day 4	129	165	134	36
3	Day 1	Day 3	104	131	110	109
4	Day 4	Day 5	146	27	22	28
5	Day 3	Day 4	80	126	33	81
6	Day 2	Day 5	136	191	143	24
7	Day 3	Day 4	92	180	63	154
8	Day 1	Day 4	148	31	7	177

Each agent communicates with the TAC server through a TCP-based agent programming interface (API) 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:[3]

- *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 8 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 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 (TT) and Shoreline Shanties (SS). TT is nicer than SS. Hotel rooms are traded in 16th price multi-unit English auctions. Overall, there are 8 hotel auctions (for each combination of hotel and night apart from the last one), which 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 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:

$$Utility = 1000 - TravelPenalty + HotelBonus + FunBonus \quad (1)$$

where $TravelPenalty = 100 * (|AD - PAD| + |DD - PDD|)$ (AD and DD are the customer's actual arrival and departure dates),

[1] Dept of Electronics and Computer Science, Southampton University, Southampton SO17 1BJ, UK.

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

[3] For full details, see *http://tac.eecs.umich.edu*.

[4] An invalid travel package receives zero utility.

HotelBonus is the bonus if the customer stays in TT, 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 1, are shown in Table 2. For example, the utility of customer 3 is obtained by the following:

$$TravelPenalty = 100 * (|AD - PAD| + |DD - PDD|) = 0,$$
$$HotelBonus = 104, \quad FunBonus = 131 + 0 + 109 = 240,$$
$$Utility = 1000 - 0 + 104 + 240 = 1344.$$

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.* 10025 below). The agent's final score is then the value of this allocation minus the cost of procuring the goods.

Table 2. SouthamptonTAC's customer allocation from game 5722. P, M, W 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	SS	P3, M4	1319
2	Day 3	Day 4	TT	P3	1263
3	Day 1	Day 3	TT	W2, M1	1344
4	Day 4	Day 5	SS	None	1000
5	Day 3	Day 4	TT	W3	1206
6	Day 2	Day 5	SS	W4, P2	1334
7	Day 3	Day 5	SS	W3, M4	1234
8	Day 1	Day 4	TT	M1	1325
				Total utility:	10025

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 (e.g., if the price of TT1 is high, the customer can change to SS1[5]). 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 paper is organised as follows. Section 2 presents the details of the SouthamptonTAC agent. Section 3 evaluates the performance of our agent. Section 4 concludes the paper.

2 AGENT DESIGN

Given the uncertainty and unpredictability involved in the TAC, there is no optimal strategy appropriate for all situations. Thus, it is desirable for the agents to be responsive to their prevailing situation during the course of bidding. To this end, Fig. 1 overviews the SouthamptonTAC agent and the remainder of the section details its behaviour.

The time period from when the agent polls the TAC server to get the most up to date asks/bids of all auctions to when it submits its bids to the TAC server is called a *round*. For SouthamptonTAC, a round lasts from 6 to 30 seconds depending on the server's load. SouthamptonTAC connects to the server in a continuous series of such rounds. In each round, our agent processes this ask/bid information in "Bids

[5] We will use the abbreviation TTn and SSn ($1 \leq n \leq 4$) for staying in the indicated hotel on a particular day.

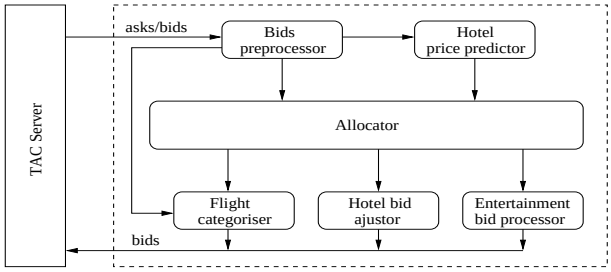

Figure 1. Overview of the SouthamptonTAC agent

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. All of this information is then input to "Allocator"[6] which calculates the optimal distribution of goods to customers given the current situation. 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 and decides whether to increase the price of its bids or to "withdraw" (to be discussed in Section 2.3) the current bids and turn to other auctions. "Flight categoriser" classifies each flight auction according to its expected change of price and based on this decides when to bid and how many trips to bid for. For example, it may delay buying the flight tickets if it believes the price change will be small, so that it has flexibility in choosing the hotel rooms. "Entertainment bid processor" determines the type and the amount of entertainment to bid for.

SouthamptonTAC divides a game into three stages: *probing* stage (up to minute 4), the *decisive* stage (from minutes 5 to 11) and the *finalisation* stage (minute 12). Hotel auctions are the most uncertain part of the game. This uncertainty stems both from the random nature of the customers' preferences and from the way opponents deal with their hotel bidding. Nevertheless, a rational agent should have submitted all its hotel bids before the end of the 4th minute (otherwise a hotel auction may close and the agent will miss out on those rooms). Thus, during the first 4 minutes the demand of the hotel market is unpredictable. Given this, SouthamptonTAC uses the probing stage to buy some flights which it has a high probability of needing, to place buy and sell bids in the entertainment auctions, and to place initial hotel bids. The agent bids for not only what it needs, but also for extra rooms in the hotels with low ask prices (since the additional outlay is comparatively small and gives the agent greater flexibility). As the decisive stage progresses, the demand of the various auctions becomes clearer and rooms are actually allocated which means the agent can more accurately decide which hotels to go for. The finalisation stage represents the agent's last chance to transact on entertainment tickets and to buy any remaining flights that are needed. There is no longer any uncertainty in this stage and so Allocator can find the optimal allocation and the appropriate bids are generated.

2.1 Flight Auctions

The flight price is perturbed every 24 to 32 seconds by a value drawn uniformly from -10 to $x(t)$. The final upper bound x (called the *flight's determinant factor*) on perturbations is a random variable chosen independently from $[10, 90]$ for each flight for each game. The upper bound on perturbations at time t is a linear interpolation

[6] Allocator uses similar linear and integer programming techniques to [2].

between 10 and the final bound x: $x(t) = 10 + (t/12:00) * (x-10)$. x is not known to the participants, however through observation of the price changes, it is possible to approximate it and to classify the flight auction into one of four categories:

$$\mathcal{F}_j = \{f \mid f\text{'s determinant factor } x \in [L_j, U_j]\} \quad (2)$$

where f represents a flight auction, L_j and U_j represent the lower and upper limits of the flight's determinant factor and $j = 0,1,2,3$: when $j = 0$, $L_0 = 10$ and $U_0 = 15$; when $j = 1$, $L_1 = 15$ and $U_1 = 30$; when $j = 2$, $L_2 = 30$ and $U_2 = 60$; and when $j = 3$, $L_3 = 60$ and $U_3 = 90$. This categorisation is computed as follows:

$$\arg \min_{0 \le j \le 3} \left| \frac{1}{n} \sum_{i=1}^{n} \delta_i - M_j \right| \quad (3)$$

where n is the number of times the price changed in the auction; δ_i is the ith price change; and M_j, called the centre of $\mathcal{F}_j$, is given by:

$$M_j = \frac{1}{U_j - L_j + 1} \left(\sum_{x=L_j}^{U_j} \left(\frac{1}{n} \sum_{k=1}^{n} \frac{(x-10)t_k}{2t_{max}} \right) \right) \quad (4)$$

where t_k is the time at which the kth price change is quoted and t_{max} is the time period of a game (720 seconds). Equation (3) computes the average price change of the flight and classifies it into the closest category. However, since the price change δ is drawn from a range, whenever δ is larger than the upper limit of the range, the flight must belong to a category with a bigger x. For each $\mathcal{F}_j$, the upper limit of the range that random changes are drawn from rises with time. Thus $U_j = g_j^u(t)$, where $g_j^u(t)$ is the upper limit of the determinant factor of $\mathcal{F}_j$ at time t. Then, suppose a flight k is currently categorised as $\mathcal{F}_j$ and the current price change is δ, if $\delta > g_j^u(t)$ and $\delta \le g_{j+1}^u(t)$, then flight k should be reclassified as $\mathcal{F}_{j+1}$.

The flight categorisation is updated in each round. Clearly as the game progresses the categorisation becomes more accurate. However, for most flights the prices rise during the game. However, if the agent buys flight tickets very early, it may fail to buy the necessary hotel rooms (leading to some invalid travel packages). Thus, what we need is a good trade-off between buying flights earlier at lower prices and buying them later to ensure they fit with the hotels that have been bought. To achieve this, our agent decides when and how many to buy of a particular flight based on the flight categorisation. It buys 8-10 tickets at the beginning of the game (the number varies in different game contexts, see Section 3). The remaining 6-8 flight tickets are left for later to ensure there is a degree of flexibility. Then, whenever an $\mathcal{F}_3$ auction is sensed, the agent will buy flights immediately. However, for an $\mathcal{F}_0$ auction, it will buy the tickets in the last minute since the price at the end of the game will be similar or less than the initial price. An $\mathcal{F}_1$ flight will be bought when the corresponding hotel rooms are guaranteed or the demand of the hotels involved with the flight is not very high. An $\mathcal{F}_2$ flight is bought immediately after the probing stage. This is because during the probing stage the expected price change is quite small (recall $g_j^u(t)$ rises with time), however after the probing stage, the increase is more significant.

2.2 Entertainment Auctions

The entertainment CDAs involve two kinds of bids: buys and sells. The entertainment bid processor handles these in the following way:

- *Number of buy bids*. Place buy bids for customer i, if there is a day without entertainment and the utility for i of entertainment of a particular kind is big (we choose 100). If a customer has multiple high entertainment preference values, place multiple buy bids on the same day to increase the chance of buying.
- *Buy bid reservation price*. Let $v_{i,j}$ be the preference valuation of customer i for entertainment j. The agent only buys a good if it can make a profit from it, *i.e.*, $v_{i,j}$ must be larger than the price of buying that good. Thus, the buy bid reservation price *bid* is given by: $bid = v_{i,j} - \psi(t)$, where $\psi(t)$ is the profit the agent can obtain if the good is transacted at *bid*. Here $\psi(t)$ is a decreasing function with time meaning that the later it is, the lower the profit the agent is willing to accept.
- *Number of sell bids*. Sell any unallocated entertainment tickets[7] and any allocated tickets that the agent can get more profit for by selling rather than by allocating to a customer.
- *Sell bid reservation price*. The reservation price *ask* is given by $ask = cost + \phi(t)$, where $\phi(t)$ is a deceasing function of time and *cost* is the preference value for an allocated ticket and a predefined value for unallocated tickets. The latter value varies according to the agent's context. If it has n unallocated tickets, the cost will be at descending prices (from 80 to 50) meaning that the more goods the agent has, the quicker it wants to sell them (thus, the sell price is lower). $\phi(t)$ decreases with time meaning the later it is, the lower the price the agent is willing to sell the good for (since selling for a small profit is better than not selling at all).

SouthamptonTAC does not wait until the ask price decreases to or the bid price rises to exactly its reservation buy/sell price. Rather, the agent continuously observes the market and when it finds an ask/bid price very close to its reservation price it will accept it. This strategy is achieved using fuzzy sets (see [1] for more details). Fig. 2 shows the fuzzy sets used to decide when to accept the current asks/bids, where θ_b and θ_s are the thresholds (set here to 0.9) of the degree that an agent would like to relax its constraints on buy/sell bids.

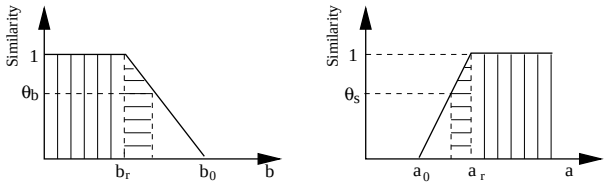

Figure 2. Fuzzy sets used in entertainment CDAs. The region with vertical lines represents the original price acceptance range. Using fuzzy sets, the acceptance range also includes the region with horizontal lines.

2.3 Hotel Auctions

Hotel auctions are the most important, uncertain and difficult part in TAC. To deal with this complexity, several strategies are used: (i) offline reasoning about hotel demand; (ii) fuzzy reasoning to predict the likely clearing prices; and (iii) "withdraw" non-profitable hotel bids.

According to the basic laws of microeconomics, the higher demand there is in a market, the higher the price of the goods. In this context, the hotels in greatest demand (determined by the offline reasoning) are TT2 and TT3 (followed by SS2 and SS3). This information is factored into the agents' reasoning about price prediction and withdrawing bids so that price information can be appropriately analysed. The hotel price predictor utilises fuzzy rules to predict the

[7] Unallocated tickets are caused by having multiple tickets for the same event or the same day for a given customer or by having no customer staying on the night of the entertainment.

clearing prices of hotels. Through observation, we find that the factors that affect the price of hotels are: (i) the ask price of that hotel auction; (ii) the counterpart hotel ask price; (iii) the counterpart hotel closing time (if applicable); (iv) the current time into the game; and (v) the rate of change of the hotel ask price. For example, in game 5960, SS3 closes first. Then the price for TT3 rises very quickly because SS3's closure means some agents fail to get SS3 and so they have to bid in TT3. Also, the rooms of day 2 have a close relationship with those of day 3 because many customers stay for successive days. Thus, the price of TT2 also rises quickly. To capture reasoning of this kind, we use the Sugeno controller [4] fuzzy reasoning mechanism (see [1] for a justification of this choice in this type of environment). The corresponding rules adhere to the following pattern:

$$\mathcal{R}_i: \; IF \; x_1 \; is \; A_{i1} \; and \; \cdots \; and \; x_n \; is \; A_{in} \; THEN \; \Delta_i = c_i.$$

where $A_{i1}, \cdots, A_{in}$ are fuzzy sets, Δ_i is a real number indicating the predicted price increase, and $c_i \in \{small, medium, big, very\_big\}$ are fuzzy parameters that are tuned to reflect the degree of competitiveness in the game (see Section 3).

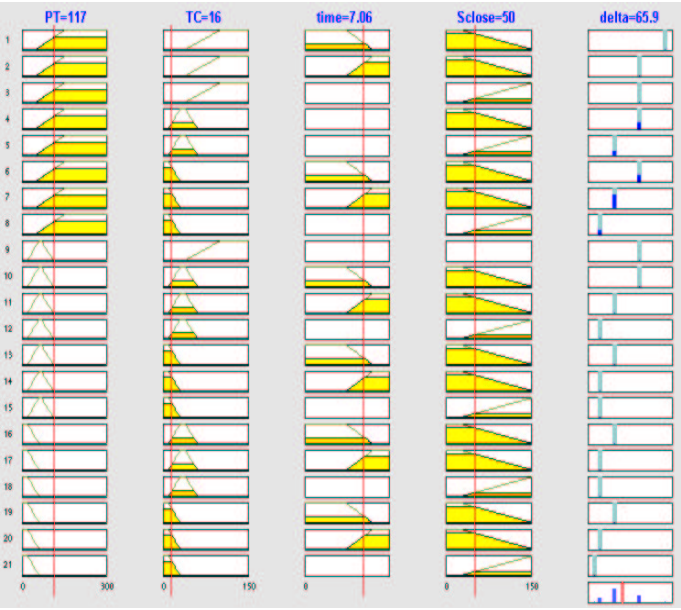

Figure 3. Fuzzy reasoning mechanism when counterpart auction closes. The rulebase is shown being applied to predicting the price of TT2. Thus, PT, TC, and Sclose correspond to PT2, TC2, and Sclose, and delta refers to TT2's predicted price increase. The current rule input state of each of these fuzzy sets is denoted by the vertical lines and the rules that are fired are those where all the fuzzy sets are matchable. Thus given the inputs shown ($PT2 = 117$, $TC2 = 16$, time=7.06, and $Sclose = 50$), rules 4-8 are fired. The fifth column represents the rule's consequent and delta gives an overall output for the rulebase at this moment in time (in this case 65.9).

Due to space limitations, we just present some typical example rules. When the agent predicts the clearing price of the TT2 auction, factors considered are (as noted above): current ask price of TT2 (PT2); current ask price for SS2 (PS2); ask price increase of TT2 in previous minute (TC2); ask price increase of SS2 in previous minute (SC2); current time (time); and time period since counterpart auction closed (Sclose). Some corresponding rules are:

- *when SS2 has not yet closed*: (i) *IF* PT2 is medium and TC2 is slow and time is early *THEN* Δ_i is medium. (ii) *IF* PT2 is low and PS2 is high *THEN* Δ_j is *medium*.
- *when SS2 has closed*: (i) *IF* PT2 is high and TC2 is quick and Sclose is short and time is early *THEN* Δ_i is *very_big*. (ii) *IF* PT2 is high and TC2 is slow and Sclose is not short *THEN* Δ_j is *small*.

In the above rules, the hotel ask price is expressed in the fuzzy linguistic terms: "high", "medium", and "low" and price change in the fuzzy sets "quick", "medium" and "slow". "short" is a fuzzy set expressing how long since the counterpart hotel auction has closed and "early" is a linguistic term expressing the current time.

Generally speaking, the prediction rules can be divided into two cases: those for when the counterpart auction has not closed and those for when it has. The former has 18 rules (of which 2 are shown above) and the latter has 21 (of which 2 are described above and all of them are shown schematically in Fig. 3). In both cases, however, the output of the rulebase is the prediction (Δ) of how much the price of the given hotel is likely to increase. This increase is then added to the current price to obtain the predicted clearing price.

Turning now to the notion of withdrawing bids. TAC does not allow hotel bid withdrawal during a game (as described in Section 1). Nevertheless our agent can effectively achieve withdrawal by the following means. Our agent decides to withdraw an auction bid either because the rooms cannot be used or because the hotel price is predicted to rise sharply. Suppose the current ask price of a hotel auction is a, if the agent wants to withdraw its current active bid it will submit a bid (for the appropriate quantity) at the price of $a+1$. In so doing, the agent believes that new bids from other agents will top its withdraw bid and thus will remove its commitment to those rooms. This method proved very effective and meant our agent could withdraw bids before the ask price rose too high.

3 EVALUATION

The TAC was made up of a preliminary round, a seeding round, the semi-finals and the final round. The preliminary round (about 250 games) determined qualification for the finals. It involved 28 agents (including a dummy agent provided by the TAC team) and for each game 8 agents were randomly drawn from the entrant pool. SouthamptonTAC was the first placed qualifier (10.88% higher than the second placed agent whitebear and 11.06% higher than the third placed agent DEFend). The seeding round determined groupings for the semi-finals. It involved about 300 games and the pool of agents consisted of most of the agents participating in the seeding round. Again SouthamptonTAC obtained the highest score (1.38% higher than the second placed agent whitebear and 2.86% higher than the third placed agent Urlaub01). The top 16 agents were organised into two "heats" for the semi-finals (consisting of 11 games) based on their position in the seeding round. In heat 1, the first 4 placed agents played games with the last 4 and in heat 2 the remaining agents played one another. The first four teams in both heats entered into the final round and the results are listed in Table 3.[8] Here SouthamptonTAC had the 3rd highest score.

Overall, in the course of the competition some 600 games were played and SouthamptonTAC had the highest mean score and lowest standard deviation (see Table 4). 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.

[8] This score was calculated without 7315, where there was a crash due to the api_call failure for SouthamptonTAC. Details can be found in http://auction2.eecs.umich.edu/tac01-scores-finals/.

Turning specifically to the final round, SouthamptonTAC obtained higher utility than both livingagents and ATTac (447 and 166 respectively per game). But our agent spent more on flights; 354 per game more than livingagents and 256 per game more than ATTac. In fact, livingagents bought all the flights at the beginning in all cases and ATTac bought 12-16 flights at the beginning, while SouthamptonTAC only bought 8-10 flights (in order to retain greater flexibility during the course of game).

Table 3. Result of Final Round (14 Oct 2001)

Rank	Agent	Score	Std Dev	Games played
1	livingagents	3670.0	622.3	24
2	ATTac	3621.6	691.6	24
3	SouthamptonTAC	3530.6	568.8	23
4	whitebear	3513.2	700.1	24
5	Urlaub01	3421.2	698.3	24
6	Retsina	3351.8	668.2	24
7	CaiserSose	3074.1	656.2	24
8	TacsMan	2859.3	1054.3	24

Table 4. Result of all games played

Rank	Agent	Average	Std Dev	Games played
1	SouthamptonTAC	3225.4	867.3	611
2	whitebear	3022.1	1027.1	620
3	TacsMan	2850.4	1113.9	624
4	Urlaub01	2834.6	1484.2	625
5	Retsina	2712.1	1496.8	617
6	ATTac	2699.4	1273.6	624
7	polimi_bot	2525.3	3285.7	513
8	umbctac	2514.3	2107.6	594
9	CaiserSose	2441.8	1899.0	620
10	PainInNEC	2385.3	1636.2	596
11	livingagents	2295.9	6506.0	607

Our post hoc analysis of the competition shows that an agent's performance depends heavily on the risk attitude of its opponents. Here a risk-averse agent (*e.g.*, CaiserSose and whitebear) corresponds to one that does not buy a large number of flight tickets at the beginning of the game and that bids according to the situation as the game progresses. This kind of agent leaves more flexibility so that it can deal with more competitive environments. In contrast, a risk-seeking agent (*e.g.*, livingagents and ATTac) 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 not cope well in environments in which the hotels are expensive. 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, hoping that the price will eventually stabilise. The consequence of this variety is that for broadly the same situation, different agents can bring about widely varying final prices. In this scheme, SouthamptonTAC is broadly risk averse.[9] Based on this observation, we identify the following types of TAC environment:

(i) *Competitive environment* (involving more than 3 risk-seeking agents). The prices of the hotels in high demand are always very high. Sometimes the prices of hotels that are not in high demand are also high. This is caused by (i) the high bid price that an agent places in the auction; (ii) the fact that some agents insist on bidding on that hotel even when the ask price becomes high; and (iii) the fact that some agents increase their bids sharply rather than gradually. For example, in game 6258, the prices of TT (SS) are (in the increasing order of day): 26 (3), 1033 (400), 801 (400), and 67 (91). 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 the agent needs to be flexible and predictive. An agent cannot buy all the flights at the very beginning of the game, otherwise, when the hotel price rises very high, it has to give up the travel package for some customers or pay very high prices for hotels. Being predictive is also important. The more accurate the predicted price, the better the outcome that can be obtained. To this end, various agents used fuzzy rule based reasoning (SouthamptonTAC), Bayesian analysis (whitebear), and machine learning (ATTac). Using such features, it is possible to obtain good scores in these games, but without it very bad scores were often obtained. In this environment, SouthamptonTAC performs very well.

(ii) *Non-competitive environment* (involving few risk seeking agents or agents with non-conflicting preferences). The prices of hotels with high demand is not high at all, and the prices for other hotels are low. For example, in game 6341, there is very little competition and the closing prices for TT (SS) are 2 (1), 2 (11), 36 (2) and 2 (1). In this situation, the best strategy is to buy all flights earlier; since the agents can always get the hotels they want. In this case, the flexible nature of SouthamptonTAC's bidding is not as effective as the risk seeking behaviour.

(iii) *Semi-competitive environment* (involving about 3 risk-seeking agents, this number varies due to the distribution of game data). There is competition, but it is not very severe. Only the hotels in high demand have a medium or high clearing price. For example, in game 6789, the clearing prices for TT (SS) are 57 (2), 132 (25), 11 (8) and 11 (4). Similar to the competitive environment, flexibility and being predictive are desirable, although risk seeking behaviour is also very effective. SouthamptonTAC does well in this situation.

In the final round, roughly 2 games were competitive, 8 were non-competitive, and 14 were semi-competitive. The respective average scores for SouthamptonTAC, livingagents and ATTac in these environment were (2898, 2547, 2660), (3786, 3942, 3964) and (3492, 3674, 3563).

4 CONCLUSIONS

SouthamptonTAC has been shown to be successful across a wide range of TAC environments; of the 600 games played in the competition it recorded the highest mean score and the lowest standard deviation. Naturally the strategies that have been employed are tailored to the specific auction context of the competition. Nevertheless we believe the TAC domain has a number of characteristics that are common to many real-world, on-line trading environments. Furthermore, we believe that the reasoning models and concepts developed for SouthamptonTAC are broadly applicable to an agent acting in a multiple auction context in which there are interdependencies.

REFERENCES

[1] M. He, H.F. Leung, and N.R. Jennings, 'A fuzzy logic based bidding strategy for autonomous agents in continuous double auctions', *IEEE Transactions on Knowledge and Data Engineering*, (2002). To appear.

[2] P. Stone et al., 'ATTac-2000: An adaptive autonomous bidding agent', *Journal of Artificial Intelligence Research*, **15**, 189–206, (2001).

[3] M. Wellman et al., 'Designing the market game for a trading agent competition', *IEEE Internet Computing*, **5**(2), 43–51, (2001).

[4] H.-J. Zimmermann, *Fuzzy Set Theory and Its Applications*, chapter 11, 203–240, Kluwer Academic Publishers, 1996.

[9] Our agent can adapt its behaviour in the direction of risk-seeking when it finds that games are not competitive.