Designing a Successful Trading Agent: A Fuzzy Set Approach

Minghua He and Nicholas R. Jennings

***Abstract*—Software agents are increasingly being used to represent humans in online 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. 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 paper describes the design, implementation and evaluation of SouthamptonTAC, one of the most successful participants in both the Second and the Third International Competitions. Our agent uses fuzzy techniques at the heart of its decision making: to make bidding decisions in the face of uncertainty, to make predictions about the likely outcomes of auctions, and to alter the agent's bidding strategy in response to the prevailing market conditions.**

***Index Terms*—Fuzzy reasoning, fuzzy set, intelligent agents, online auctions, trading agent competition (TAC).**

I. Introduction

AGENT-MEDIATED electronic commerce involves software agents acting on behalf of some or all the parties in e-commerce transactions [20], [22], [33], [42]. Here, a software agent is viewed as an encapsulated computer system, situated in some environment, that is capable of flexible autonomous action in that environment in order to meet its design objectives [25]. The rationale for introducing such agents in e-commerce scenarios is to offer faster, cheaper, more convenient, and more agile ways for both customers and suppliers to trade. As the agents represent distinct stakeholders and organizations, the *de facto* way in which they interact is through some form of *negotiation*. In human negotiations, two or more parties typically bargain with one another to determine the price or other transaction terms [11]. In automated negotiations, the agents prepare bids and evaluate offers in order to obtain the maximum return for the parties they represent [26]. Such automated negotiation leads to dynamic pricing which ensures that goods and services are allocated to the entity that values them most highly [20]. This, in turn, relieves the merchant of the burden of *a priori* fixing the price. In contrast to many human negotiations, automated negotiation can be very fast since decisions and exchanges can occur rapidly. Automated negotiation can also remove the human sensibilities that are often associated with bargaining. Moreover, complicated negotiation problems (e.g., those involving many inter-related goods) are often too difficult for consumers to handle manually. In such cases, automated negotiation systems can help ordinary users perform like professional negotiators. In fact, we believe that in complex settings software agents are likely to be more effective than human bidders (preliminary empirical evidence to substantiate this hypothesis is given in [9]).

Against this background, auctions are one of the most widely studied and employed negotiation protocols in e-commerce today [36]. For example, in the Internet Auction List[1] there are currently more than 2500 auction company listings (as of 2003). Auctions are so popular because they are a very efficient and effective method of allocating goods or services in dynamic situations [50]. Moreover, online auctions make the physical limitations of traditional auctions disappear (e.g., time, space and presence), and they provide millions of globally dispersed customers with more varieties of goods that can be selected within a flexible pricing mechanism [3]. Although there are millions of different auction protocols [51], four types of single-sided auction[2] are common [48]: English (first-price ascending), First-price sealed-bid, Vickrey (second-price sealed-bid), and Dutch (first-price descending). The most common type of double-sided auction is the continuous double auction (CDA) [13] which allows buyers and sellers to continuously update their bids or asks at any time in the trading period. Moreover, coupling the accessibility of online auctions with the capabilities of software agents also opens up the possibility of competing in multiple auctions simultaneously (either for the same good or for inter-related goods). Such a strategy has several advantages over participating in single auctions; for example, it can increase the chance of getting the good for customers, bring greater profit to customers by comparing multiple auctions and transacting at the cheapest price, and make the auction markets themselves more efficient by ensuring the transaction price is close to the equilibrium price [39].

Given the potential and the importance of using agents in online auction settings, there has been considerable research endeavour in developing bidding strategies for different types of agents in different types of auctions (see Section V for more details). Therefore, in order to develop a means of comparing and evaluating this work, it was decided to establish an International

Manuscript received January 27, 2003; revised August 27, 2003.

The authors are with the School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K. (e-mail: mh@ecs.soton.ac.uk; nrj@ecs.soton.ac.uk).

Digital Object Identifier 10.1109/TFUZZ.2004.825064

[1]http://www.internetauctionlist.com

[2]In single-sided auctions, only buyers or sellers can submit bids or asks. This contrasts with double-sided auctions in which buyers can submit bids and sellers can submit asks.

Trading Agent Competition (TAC) (similar in spirit to other initiatives such as RoboCup,[3] RoboCupRescue[4] and the Planning Competition[5]). In this competition, software agents compete against one another in 28 simultaneous auctions in order to procure travel packages (flights, hotels and entertainment) for a number of customers (see Section II for more details of the roles).

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 in the TAC involves uncertainty caused by the random features of the game, the opponents' strategies and the particular combination of opponents. Against this background, this paper reports upon the design and implementation of our particular trading agent (called **SouthamptonTAC**) which participated in both the competition in 2001 (TAC-01) and in 2002 (TAC-02).

SouthamptonTAC was one of the most successful agents in both competitions (see Section IV) and this paper details its design and implementation and evaluates when and why it is successful. In more detail, SouthamptonTAC is an adaptive agent that varies its bidding strategy according to its perception of the prevailing market conditions. It uses fuzzy reasoning techniques to predict closing prices of the auctions, fuzzy recognition to assess the degree of competitiveness in the prevailing market context, and fuzzy set technique to control bidding behavior. The decision problem in a TAC game aims to ensure the agent gets the maximum utility at the lowest cost. This depends on many factors given the current environment, such as the flights bought, the price of the hotel auction, the customers' preferences, the hotel rooms the agent currently holds and so on. These factors are usually highly ambiguous. Therefore, fuzzy set theory provides a good way to handle them. We chose to adopt a fuzzy logic based approach, in particular, because we wish to develop a practicable agent that can cope with the uncertainties in a timely manner and fuzzy techniques have proven to be successful in a wide range of domains with these characteristics (e.g., fuzzy control to drive car-like vehicles [12], making medical diagnosis [53], vehicle dispatching [46], and emergency electric power distribution [35]). See Section V for details of the other alternatives we considered.

This work advances the state of the art in two main ways. In terms of the TAC itself, we developed novel reasoning models and prediction methods that enable an agent to bid across multiple heterogeneous auctions that have interdependencies. Through the competition and our systematic evaluation this reasoning mechanism is shown to be both highly effective and practical (it has to operate in a time-constrained environment and has to cope with the uncertainty of operating over the Internet with its concomitant latency problems). In more general terms, we believe that a number of the technologies we developed can be used in other complex auction settings and our insights and experiences about building a successful trading agent will also transfer (see the discussion in Section VI for more details).

The remainder of the paper is organized as follows. Section II describes the trading agent competition. Section III presents the details of the SouthamptonTAC agent. Section IV evaluates the performance of SouthamptonTAC. Section V discusses the related work. Finally, Section VI concludes this paper and outlines the areas of future work.

TABLE I
SOUTHAMPTONTAC'S CUSTOMER PREFERENCES FOR GAME TAC-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

II. Trading Agent Competition

TAC-01 and TAC-02 involved 27 and 26 agents, respectively, developed by universities and research labs from around the world [18], [49]. In each 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.[6] A valid travel package for an individual customer consists of: i) a round trip flight during a five-day period (between TACtown and Tampa), and ii) a stay at the same hotel[7] 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 maximize 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 min and involves 28 auctions. Each of the three good types are traded in an auction with different rules.[8]

- *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

[3]An international project that uses soccer as a central topic; see http://www.robocup.org for more details.

[4]RoboCupRescue is a new research domain which targets search and rescue in large scale disasters (such as earthquakes); see http://www.r.cs.kobe-u.ac.jp/robocup-rescue/ for more details.

[5]The International Planning Competition aims to provide a forum for empirical comparison of planning systems; see http://www.dur.ac.uk/d.p.long/competition.html for more details.

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

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

[8]For full details, see http://www.sics.se/tac.

ask prices update randomly, every 24–32 s, 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.

- *Hotels*. There are two hotels: Tampa Towers (T) and Shoreline Shanties (S). T is nicer than S. Hotel rooms are traded in 16th price multiunit 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 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 CDAs. Overall, there are 12 CDAs (for each kind of entertainment for each of days 1–4). 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.

By means of illustration, Table II gives the market running state of all the auctions at a single moment in time of the game in tac-5722. A customer's utility from a valid travel and entertainment package[9] is given by

$$\text{Utility} = 1000 - \text{TravelPenalty} + \text{HotelBonus} + \text{FunBonus}$$

where $\text{TravelPenalty} = 100 * (|\text{AD} - P\text{AD}| + |\text{DD} - P\text{DD}|)$ (here, 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 III. For example, the utility of customer 3 is obtained by the following:

$$\begin{aligned}\text{TravelPenalty} &= 100 * (|\text{AD} - P\text{AD}| \\ &\qquad + |\text{DD} - P\text{DD}|) = 0\\ \text{HotelBonus} &= 104\\ \text{FunBonus} &= 131 + 0 + 109 = 240\\ \text{Utility} &= 1000 - 0 + 104 + 240 = 1344.\end{aligned}$$

[9]An invalid travel package receives zero utility.

TABLE II
MARKET STATE OF ALL AUCTIONS IN GAME TAC-5722. THE REMAINING TIME IN THE GAME IS 6 MIN 42 S

Auction	Bid Quote	Ask Quote	Status
Alligator Wrestling on day 1	1.1	86	running
Alligator Wrestling on day 2	70	150	running
Alligator Wrestling on day 3	65	139.38	running
Alligator Wrestling on day 4	66	107.51	running
Amusement Park on day 1	66	70	running
Amusement Park on day 2	66	90	running
Amusement Park on day 3	84	119	running
Amusement Park on day 4	50	89.74	running
Museum on day 1	50	99	running
Museum on day 2	50	146.73	running
Museum on day 3	50	175	running
Museum on day 4	50	80	running
Inflight day 1	–	328	running
Inflight day 2	–	337	running
Inflight day 3	–	381	running
Inflight day 4	–	299	running
Outflight day 2	–	400	running
Outflight day 3	–	370	running
Outflight day 4	–	434	running
Outflight day 5	–	270	running
Tampa Towers Hotel day 1	0	22.35	closed
Tampa Towers Hotel day 2	0	59.49	running
Tampa Towers Hotel day 3	0	33.60	running
Tampa Towers Hotel day 4	101	101.5	running
Shoreline Shanty day 1	0	0	running
Shoreline Shanty day 2	33	42	running
Shoreline Shanty day 3	22.2	29	running
Shoreline Shanty day 4	1	1	closed

TABLE III
SOUTHAMPTONTAC'S CUSTOMER ALLOCATION FROM GAME TAC-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	S	P3, M4	1319
2	Day 3	Day 4	T	P3	1263
3	Day 1	Day 3	T	W2, M1	1344
4	Day 4	Day 5	S	None	1000
5	Day 3	Day 4	T	W3	1206
6	Day 2	Day 5	S	W4, P2	1334
7	Day 3	Day 5	S	W3, M4	1234
8	Day 1	Day 4	T	M1	1325
				Total utility:	10025

TABLE IV
EXPENDITURE FOR SOUTHAMPTONTAC IN TAC-5722. A NEGATIVE NUMBER MEANS THE AGENT OBTAINS THE INDICATED AMOUNT OF UTILITY BY SELLING THE GOOD

Good	Day 1	Day 2	Day 3	Day 4	Day 5	Total
AlligatorWrestling	0	1 at 91.9	1 at 108.9 1 at 100.5	1 at 120.5	0	421.80
AmusementPark	0	-2 at 80 -1 at 139.5	0	0	0	-299.5
Museum	-1 at 89	0	0	0	0	-89
Inflight	1 at 282 1 at 340	1 at 314	3 at 341 1 at 381	1 at 277	0	2617
Outflight	0	0	1 at 390	2 at 345 1 at 415	2 at 272 1 at 273	2585
TampaTowers Hotel	2 at 22.35	2 at 120	3 at 33.6	0	0	385.5
ShorelineShanty Hotel	4 at 1	1 at 47.5	3 at 30	5 at 1	0	146.5
					Total cost:	5767.3

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., 10 025). The agent's final score is then the value of this allocation minus the cost of procuring the goods. For example, the agent's cost of obtaining the goods in game tac-5722 is shown in Table IV. Thus, the score in this game is $10\,025 - 5767.30 = 4257.70$.

Designing a bidding strategy for the TAC auction context is a challenging problem.

- *There are interdependencies between auctions*. That is, what goods to buy and how many to buy in one auction are relative to the progress of other auctions. These interdependencies exist between different kinds of auctions. For example, flights will be useless if the hotel rooms are not available and if no customer stays in Tampa on a particular day, the entertainment ticket on that day will be useless. The interdependence also exists between different dates within the same kind of auction. For example, customers must stay in the same hotel during their stay. Thus if a customer stays in T1,[10] the agent will also need to bid in the auctions of other days for T. Finally, interdependencies exist between the auctions of the same day, same kind counterpart auctions.[11] For example, if the price of T1 is high, the customer can change to S1.
- *The bidding involves uncertainty*. For example, flight prices start randomly and change continuously in a random fashion and one randomly selected hotel auction closes from the 4th–11th minute.
- *The bidding involves tradeoffs*. 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, while the flight prices may be high if it buys the flights later.

III. SOUTHAMPTONTAC

Our agent design for TAC-01 and TAC-02 is broadly similar. However, since SouthamptonTAC-02 built upon our experiences in TAC-01 this is the agent we describe in this section. The main differences between the two are as follows.

- SouthamptonTAC-02 is an adaptive agent that varies its bidding strategy according to its perception of the prevailing market conditions (see Section III-C for details). SouthamptonTAC-01 had a fixed strategy that it used in all contexts.
- SouthamptonTAC-02 does hotel closing price prediction differently. Although they both use the same basic technique, SouthamptonTAC-02 has two rule bases for predicting prices when the counterpart auction has closed (one for when it has just closed and one for when it has been closed for a longer period of time). SouthamptonTAC-01 only had one such rule base (see Section III-G-I for more details).

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

[11]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.

We now deal, in turn, with each of the main components of our agent.

A. Classifying TAC Environments

Our *post hoc* analysis of the TAC-01 competition [23] 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 (see later). 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) (see Section IV-B). 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 stabilise (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 [23], 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).

B. Agent Architecture

Given the uncertainty in the TAC, it is desirable for the agents to be responsive to their prevailing situation during the course

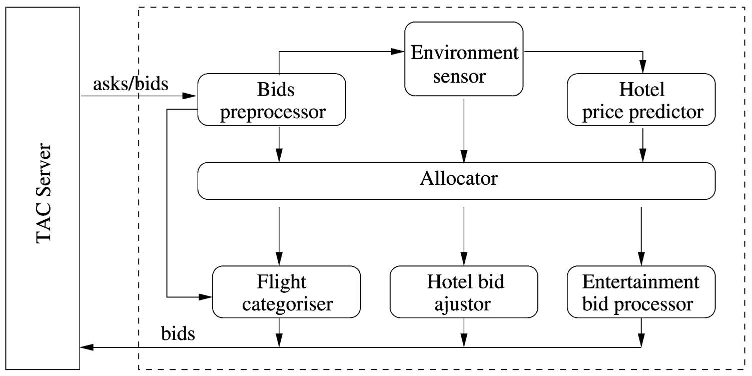


Fig. 1. Overview of the SouthamptonTAC agent.

of bidding. To this end, Fig. 1 overviews the SouthamptonTAC agent. 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*. SouthamptonTAC connects to the server in a continuous series of such rounds (the length of a round depends on the location of the TAC server as well as the server load, but it typically varies between 2–30 s). In each round, the agent first processes this ask or bid information in Bids Preprocessor to get the prices of different goods, number of goods it actually owns and may possibly own (only for hotel rooms) and its current active bids. Then, Hotel Price Predictor is used to predict the likely clearing price of each hotel auction (see Section III-G1). All of this information is then input to Allocator which calculates the optimal distribution of goods to customers given the current situation (see Section III-D for more details). Given this assignment, the agent then determines its subsequent bidding actions. Flight Categoriser uses updated flight prices to classify each flight auction according to its expected change of price and takes the output of Allocator to determine how many trips to bid for (see Section III-E). 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. 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" (see Section III-G2) the current bids and turn to other auctions (see Section III-G3). Entertainment Bid Processor determines the type and the amount of entertainment tickets to bid for (see Section III-F). Where SouthamptonTAC differs from its predecessor is in having an Environment Sensor in the architecture. This component (described in more detail in Section III-C) aims to determine what type of environment the agent is presently situated in (as detailed in Section III-A). The reason for doing this is so that the agent can adapt its bidding strategy accordingly (see Section III-H).

Among these components, the most important ones are the Environment Sensor, Hotel Price Predictor, and Allocator. The Environment Sensor aims to determine the degree of competitiveness in the environment both before a game starts and during the course of a game. It does this because the agent will use different bidding strategies in the different situations. The Hotel Price Predictor is important because it lets the agent plan about how best to draw up travel plans for the customers. The Allocator is important because it can allocate the goods the agent has bought to its customers in an optimal way and because it highlights what goods still need to be bought.

SouthamptonTAC divides a game into three stages: the *probing* stage (up to minute 4), the *decisive* stage (from minutes 5 to 11) and the *finalization* 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 min the demand of the hotel market is unpredictable. Given this, SouthamptonTAC uses the probing stage to buy some flights which it has a high possibility 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 finalization 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.

C. 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 [37] 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 ten games[12] from the price history and uses the maximum average price as a reference price to classify the environment in a given

[12]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.

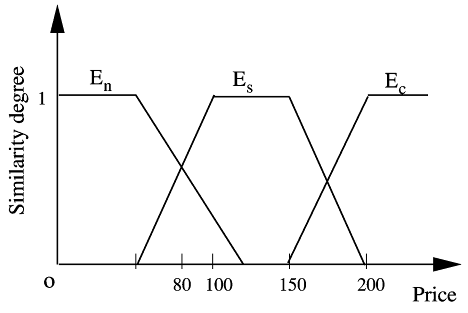


Fig. 2. Fuzzy sets of the three environment types.

game. We use the maximum price in this fashion since if one price is high it is likely 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 $T_1,\ldots,T_4$ and $S_1,\ldots,S_4$ are $P_{T_1},\ldots,P_{T_4}$ and $P_{S_1},\ldots,P_{S_4}$. 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, semicompetitive and noncompetitive environments respectively (see Fig. 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

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

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:

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

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

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

D. Allocator

The Allocator component of our agent operates in two different modes: Allocator1 deals with the allocation of available and unavailable goods and outputs what flights and hotels to buy during the first 11 min of a game and Allocator2 does the same for flights and entertainment tickets in the final minute when the hotel situation is finalised. By means of illustration, a sample output from the Allocator is shown in the equation at the bottom of the page. This indicates the agent needs to buy one inflight ticket on day 1, three on day 2, and two on days 3 and 4; buy two outflight tickets on day 3, and three on days 4 and 5; that good hotels are needed for day 2 (three rooms), day 3 (3 rooms) and day 4 (three rooms); that bad hotels are one room); and that the plans for the individual customers are currently as follows: Customer 1 goes on day 4, returns on day 5 and will stay in the good hotel (1 in third element of tuple), customer 2 will go on day 1, return on day 4, stay in the bad hotel (0 in third element of tuple), go to wrestling on day 3, the amusement park on day 2 and the museum on day 1; and so on for each of the remaining customers. Both Allocator1 and Allocator2 use the linear programming package *LPsolver*[13] to generate their optimal solutions (it never took more than 1 s on a 1.33 GHz Pentium during all games played).

Dealing first with Allocator1, in total this has 92 constraints and 272 variables.[14] In more detail, each customer can choose from inflight day (1–4) and outflight day (2–5) and hotel type (T or S). This means that in total there are 20 valid packages for each customer (see Appendix A for more details). Given 8 customers, this leaves 160 combinations (160 variables in allocators). For each customer, the agent can choose one from 12 entertainment tickets (three types for 4 days). Thus, there are 96 variables for eight customers. Moreover, there are eight variables to denote the number of flight tickets needed for each flight auction and eight variables to denote the hotel rooms needed for each hotel auction. Since from the 4th minute, one hotel auction closes, the number of variables decreases from 272 to 264 at the end of the 11th minute. The 92 constraints come from the fact that: each customer only gets one valid package (eight constraints), flight tickets/hotel rooms/entertainment tickets allocated must be less than the number the agent has or is going to buy (28 constraints), each customer can only use each type of entertainment ticket once (24 constraints), and

[13]There are other methods that are suitable to model the problem. For example, the problem could be formalised as a multiobjective problem using vectors [41]. However, the *LPsolver* is shown to find the solution very quickly and is easy to use. *LPsolver* is based on lp_solve, a simplex-based code for linear and integer programming problems by M. Berkelaar. The source is available at ftp://ftp.es.ele.tue.nl/pub/lp_solve/lp_solve.tar.gz.

[14]Our approach is based on that used in ATTac-2000 [44] but we improve upon their method. We use only 92 constraints while ATTAC-2000 has 188 constraints. Thus our Allocator greatly improves the speed of finding a solution.

$$\begin{array}{llll} \text{buy inflight} : (1,3,2,2) & \text{buy outflight} : (0,2,3,3) & \text{buy good hotel} : (0,3,3,3) & \text{buy bad hotel} : (1,1,1,0) \\ \text{customer 1} : (4,5,1,0,0,0) & \text{customer 2} : (1,4,0,3,2,1) & \text{customer 3} : (3,5,1,3,0,0) & \text{customer 4} : (3,4,1,3,0,0) \\ \text{customer 5} : (4,5,1,0,0,0) & \text{customer 6} : (2,3,1,0,0,0) & \text{customer 7} : (2,3,1,0,0,0) & \text{customer 8} : (2,4,1,3,2,0). \end{array}$$

each customer must use its entertainment tickets between the days he stays in TACtown (32 constraints).

The Allocator2 deals with the allocation of available and unavailable goods and outputs what flights and entertainments to buy in the last minute (this solver has 276 variables and 92 constraints). The first 264 variables are the same as before, except that the last 12 variables denote the entertainment ticket numbers for each type each day (12 variables). The constraints are the same as those shown previously.

E. Flight Auctions

The flight price is perturbed every 24–32 s 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 between 10 and the determinant factor x

$$x(t) = 10 + \frac{t \times (x - 10)}{t_{\max}} \tag{1}$$

where $t_{\max}$ is the time period of a game (720 s) and x is not known to the participants. Fig. 3 shows four different categories of flights in game tac2-7011 and leads to a number of observations. First, during the first 4 min of the game, the distribution of the prices vary in a small range. This is because the current time is early and time is the dominant factor determining the range of prices according to (1). Second, the price changes increase or decrease gradually during the game since time is continuously changing. Third, at the end of each game, some flight prices are lower than their initial price (Category 0), some rise slowly (Category 1), some rise quickly (Category 2) and some rise very rapidly (Category 3). Here, the differences are mainly due to the different final bound values of x.

Given these observations, we believe it is important to try and classify the various flight auctions at run-time since this categorization should lead to different bidding behavior. To this end, our agent observes the changes in prices and puts each flight auction into one of four categories

$$\mathcal{F}_j = \{f \mid f's \text{ determinant factor } x \in [L_j, U_j]\} \\ (j = 0, 1, 2, 3) \tag{2}$$

where f represents a flight auction, and L_j and U_j are the lower and upper limits of the flight's determinant factor, respectively ($L_0 = 10$ and $U_0 = 15$; $L_1 = 15$ and $U_1 = 30$; $L_2 = 30$ and $U_2 = 60$; and $L_3 = 60$ and $U_3 = 90$).[15] We found that although the increase or decrease is randomly drawn, if x is small, the price does not rise quickly; conversely, if x is high, the price will rise rapidly. The agent computes its increase or decrease so far and classifies each flight. We believe it is unnecessary to find the precise x because even though the increase or decrease is randomly drawn an x that is close to the real "x" is sufficient to approximate the expected range of change. This categorization is computed as follows:

[15]These values were first picked based on our experience with the games. Then, they were tested by a large number of games and shown to produce reasonable classifications.

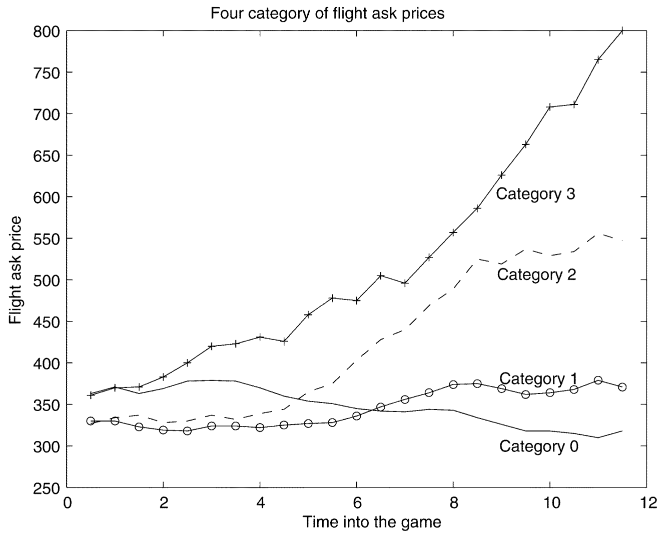


Fig. 3. Four categories of flight ask prices.

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

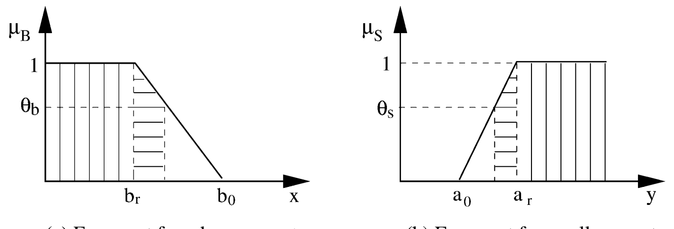
where n is the number of times the price changed in the auction; δ_i is the ith price change; and M_j, called the center of $\mathcal{F}_j$, is given by

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

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 s).

Formula (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 categorized 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 categorization is updated in each round. Clearly, as the game progresses the categorization becomes more accurate (see Section IV-C). 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 tradeoff 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 first needs to decide what type of TAC environment it is situated in (see Section III-C). For noncompetitive games, the agent



(a) Fuzzy set for a buyer agent (b) Fuzzy set for a seller agent

Fig. 4. Fuzzy sets used in entertainment CDAs. Here b_0/a_0 is the current highest bid/lowest ask; b_r/a_r is the agent's reservation price and θ_b/θ_s is the agent's threshold of accepting the ask/bid. The region with vertical lines represents the original price acceptance range. Using fuzzy sets, the acceptance range also includes the region with horizontal lines.

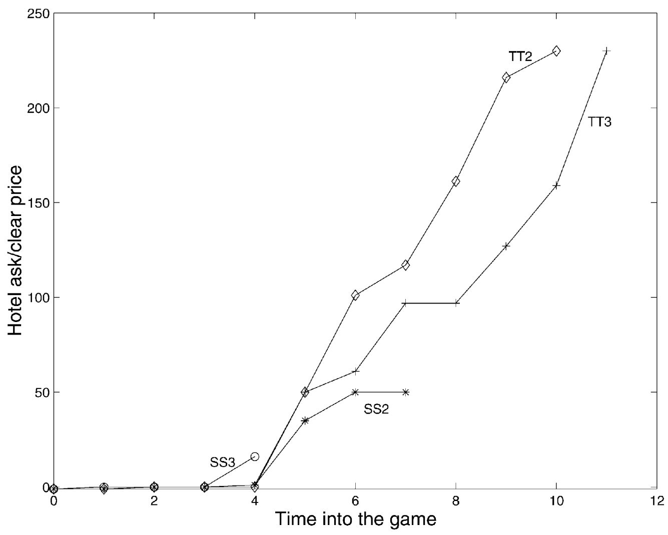


Fig. 5. Hotel prices changes in game TAC2-5960.

will buy all the flight tickets once when the game starts. For competitive or semicompetitive environments, the agent buys a number of flight tickets at the beginning of the game (8–10 for competitive and 12–14 for semicompetitive games) because it knows it will need some flights. Then, when and how many of the remaining ones to buy of a particular flight is based on the flight categorization. This delay in buying flight tickets ensures 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 (see discussion in Section III-H for the rationale).

F. Entertainment Auctions

The entertainment CDAs involve two kinds of bids: buys and sells. That is, an agent can place both bids to buy and asks to sell. It determines the amount of bids or asks to place as well as the price of bids or asks. The Entertainment Bid Processor handles entertainment bidding. A buy (sell) bid will immediately match the lowest price standing sell (highest price standing buy) bid that has a price at or below (above) the price of the buy (sell) bid. Bids match at the price of the standing bid in the auction.

- *Number of buy bids.* Place buy bids for a particular customer so that the allocation of the entertainment tickets for that customer is maximally satisfactory. For example,

TABLE V
FUZZY RULE BASE WHEN COUNTERPART AUCTION IS OPEN

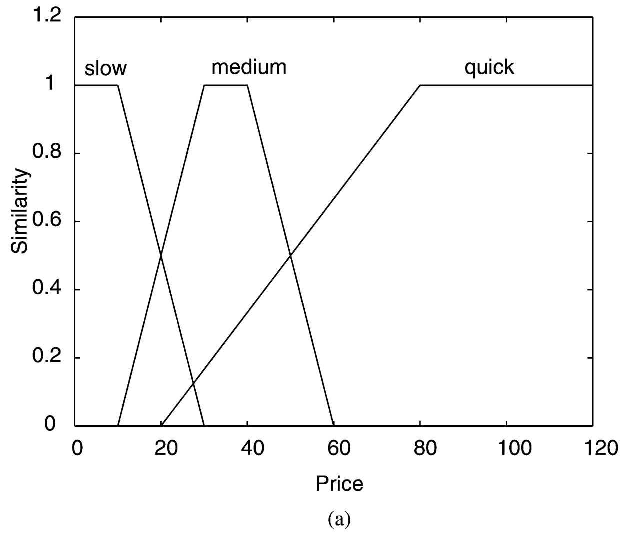
IF *P* is *high* and *C* is *quick* THEN Δ is *big*.
IF *P* is *high* and *CP* is *high* and *C* is *not-quick* THEN Δ is *big*.
IF *P* is *high* and *CP* is *not-high* and *C* is *not-quick* THEN Δ is *medium*.
IF *P* is *low* and *CP* is *high* THEN Δ is *medium*.
IF *P* is *low* and *CP* is *not-high* 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 VI
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 VII
FUZZY RULE BASE WHEN COUNTERPART AUCTION HAS CLOSED FOR MORE THAN 1 MIN

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*.



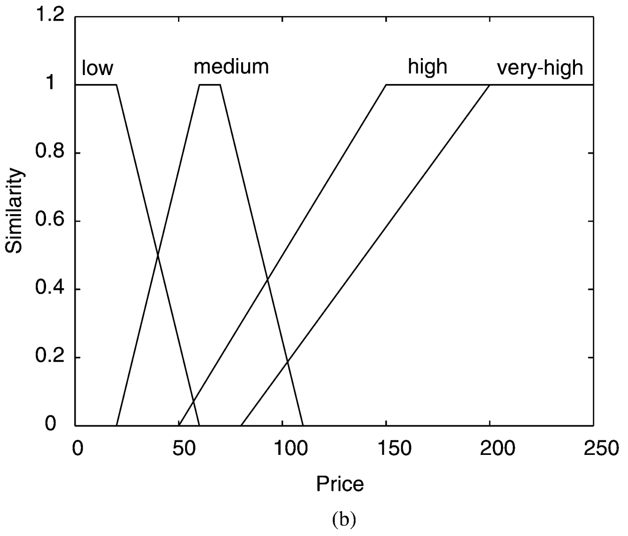


Fig. 6. (a) Fuzzy sets of price change. (b) Fuzzy sets for price of Tampa Towers/Shoreline Shanties (T/S).

if a customer will stay for one day, the agent will buy the tickets with the highest preference value for that day; if the customer stays for two days, buy tickets with the two highest preference values for each day, and so on. Here, the agent places extra bids to increase the chance of obtaining a ticket. Whenever an agent is successful in buying a particular ticket for a given day, it withdraws its buy bids for that entertainment on the other days.

- *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: $bid = v_{i,j} - \psi(t)(\psi(t) > 0)$, 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[16] 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 $\mathrm{ask} = \mathrm{cost} + \phi(t)(\phi(t) > 0)$, where $\phi(t)$ is a decreasing function of time and cost is the preference value for an allocated ticket and a predefined value for unallocated ones. 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).

[16]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.

SouthamptonTAC does not wait until the ask price decreases to or the bid price rises to exactly its reservation buy or sell price. Rather, the agent continuously observes the market and when it finds an ask or bid price very close to its reservation price it will decrease or increase its ask or bid to match it. For example, if a bid is 79.5 and the ask price of our agent is 80, it evaluates the bid and decides to decrease the ask bid to 79.5 so as to accept the bid. This strategy is achieved using fuzzy sets. The strategy is simple but effective since it avoids missing transactions where the bid and ask are quite close. Fig. 4 shows the fuzzy sets used to decide when to accept the current asks or bids, where θ_b and θ_s are the thresholds of the degree that an agent would like to relax its constraints on buy or sell bids.[17] Suppose two fuzzy sets B and S are characterized by the membership function $\mu_B : X \to [0,1]$ and $\mu_S : Y \to [0,1]$. $\mu_B(x)$ and $\mu_S(y)$ are interpreted as the degree of membership of x, (i.e., current asks placed by sellers) in fuzzy set B for each $x \in X$ and y, (i.e., current bids placed by buyers) in fuzzy set S for each $y \in Y$ respectively. Formulas (5) and (6) are the similarity membership functions for bids and asks.

$$\mu_B(x) = \begin{cases} 1, & \text{if } x < b_r \\ 0, & \text{if } x > b_0 \\ \frac{b_0 - x}{b_0 - b_r}, & \text{if } b_r \le x \le b_0 \end{cases} \tag{5}$$

$$\mu_S(y) = \begin{cases} 1, & \text{if } y > a_r \\ 0, & \text{if } y < a_0 \\ \frac{y - a_0}{a_r - a_0}, & \text{if } a_0 \le y \le a_r. \end{cases} \tag{6}$$

G. Hotel Auctions

Hotel auctions are the most important, uncertain and difficult part in TAC. Since customers can have no entertainment and flights tickets are available throughout the game, only hotel auctions are uncertain. Moreover, failure in a single hotel auction can cause the failure of an entire travel package. To deal

[17]The main elements that should be adopted as reference in the choice of the threshold could be the time into the game. With the progress of the game, the agent is more likely to relax $\theta_b(\theta_s)$ in order to buy (sell) a good. In addition, the risk attitude of the agent also has relation to the value of this threshold value.

TABLE VIII
SIMILARITY FUNCTIONS OF FUZZY SETS FOR PRICE OF T/S

$\mu_{low}(x) = \begin{cases} 1 & \text{if } x < 20 \\ 0 & \text{if } x > 60 \\ \frac{60-x}{40} & \text{if } 20 \leq x \leq 60 \end{cases}$	$\mu_{medium}(x) = \begin{cases} 1 & \text{if } 60 \leq x \leq 70 \\ 0 & \text{if } x < 20 \text{ or } x > 110 \\ \frac{x-20}{40} & \text{if } 20 \leq x \leq 60 \\ \frac{110-x}{40} & \text{if } 70 \leq x \leq 110 \end{cases}$	$\mu_{high}(x) = \begin{cases} 0 & \text{if } x < 50 \\ 1 & \text{if } x > 150 \\ \frac{x-50}{100} & \text{if } 50 \leq x \leq 150 \end{cases}$

TABLE IX
SIMILARITY FUNCTIONS FOR FUZZY SETS OF PRICE CHANGE

$\mu_{slow}(x) = \begin{cases} 1 & \text{if } x < 10 \\ 0 & \text{if } x > 30 \\ \frac{30-x}{20} & \text{if } 10 \leq x \leq 30 \end{cases}$	$\mu_{medium}(x) = \begin{cases} 1 & \text{if } 30 \leq x \leq 40 \\ 0 & \text{if } x > 60 \text{ or } x < 10 \\ \frac{x-10}{20} & \text{if } 10 \leq x \leq 30 \\ \frac{60-x}{20} & \text{if } 40 \leq x \leq 60 \end{cases}$	$\mu_{quick}(x) = \begin{cases} 0 & \text{if } x < 20 \\ 1 & \text{if } x > 80 \\ \frac{x-20}{60} & \text{if } 20 \leq x \leq 80 \end{cases}$

with this complexity, several strategies are used: i) fuzzy reasoning to predict the likely clearing prices (see Section IV-C for an evaluation of the prediction method); ii) "withdraw" nonprofitable hotel bids; and iii) reasoning to determine when to switch between bidding for the different hotels. Each of them are dealt with in turn.

1) Fuzzy Reasoning About Hotel Closing Prices: According to the basic laws of supply and demand theory [38], the higher demand there is in a market, the higher the price of the goods. Thus the competition among the agents on a particular hotel auction leads to a rise in the price of the hotels. For example, T2 and T3 are in greatest demand, since staying in the good hotel gets the higher utility and day 2 and day 3 are part of most customers' stays. Therefore, their prices are always the highest. This information, as well as the price changes, are factored into the agent's reasoning about price prediction. The reasoning utilizes fuzzy rules to predict the clearing prices of hotels. The motivation for using fuzzy rules is based on our positive experiences of their effectiveness in other auction contexts [24], as well as their use in other domains where there is a need to cope with uncertainties.[18] Through observation, we find that the factors that effect the price of hotels 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). Fig. 5 shows the relation between the closing of several hotels in game tac2-5960. Here, S3 closes first (at the 4th minute). Then, the price for T3 rises very quickly because S3s closure means some agents fail to get S3 and so they have to bid in T3. Usually, the price of S2 is high, but since S2 closes early, its price is relatively low. 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 T2 also rises quickly.

To capture reasoning of this kind, we use the Sugeno controller[19] [45], [56], since it is easy to use and has already shown to be effective in [24]. In more detail, the fuzzy reasoning inference mechanism employed here adheres to the fuzzy reasoning pattern shown in the equation at the bottom of the page, where $A_{11}, \ldots, A_{nm}$ are fuzzy sets, and $\Delta_1, \ldots, \Delta_n$ are real numbers indicating the predicted price increase. The output of the individual rule is denoted as Δ_i and $c_i \in \{\text{small}, \text{medium}, \text{big}, \text{very-}big\}$ $(i \in \{1, \ldots, n\})$ are fuzzy parameters predefined by the designer.[20]

The firing level α_i of the rules $\mathcal{R}_i$ is computed by the Min operator. That is,

$$\alpha_i = \min\{A_{i1}(x'_1), \ldots, A_{im}(x'_m)\} \tag{7}$$

where $A_{i1}(x'_1), \ldots, A_{im}(x'_m)$ are the membership functions of the corresponding fuzzy sets A_{i1} and A_{im}, respectively. According to the Sugeno controller's definition, the crisp control action (i.e., the predicted increase of hotel closing price) of the rule base is obtained by

$$\Delta' = \frac{\sum_{i=1}^{n} \alpha_i \Delta_i}{\sum_{i=1}^{n} \alpha_i}. \tag{8}$$

[18]Other methods could also have been applicable in this context (see [31] for a survey of uncertainty techniques in agent systems). However, fuzzy techniques have proven to be successful in a wide range of practical domains where there is a need to cope with uncertainties in a timely manner (see [12], [46], [53], and [54] for some examples).

[19]The use of other fuzzy logic controllers, such as the conventional Mamdani controller [56] is also possible, and such of controllers could improve the performance of our algorithms still further.

[20]These parameters are chosen based on the small or big values in the game history and adapted based on the experience of the designer through experiments.

$$\begin{array}{ll}
\mathcal{R}_1: & \text{if } x_1 \text{ is } A_{11} \text{ and} \ldots \text{ and } x_m \text{ is } A_{1m} \text{ then } \Delta_1 = c_1 \\
\textbf{also} & \\
\vdots & \\
\textbf{also} & \\
\mathcal{R}_n: & \text{if } x_1 \text{ is } A_{n1} \text{ and} \ldots \text{ and } x_m \text{ is } A_{nm} \text{ then } \Delta_n = c_n \\
\textbf{fact}: & x_1 \text{ is } x'_1 \text{ and} \ldots \text{and } x_m \text{ is } x'_m \\
\hline
\textbf{consequence}: & \Delta'
\end{array}$$

TABLE X
RESULT OF TAC-01

(a) TAC-01 seeding round.

Rank	Agent	Avg(-10 worst)	Std Dev	Games
1	SouthamptonTAC	3163.8	855.3	315
2	whitebear	3163.8-43	881.4	318
3	Urlaub01	3163.8-88.3	1197.8	319
4	livingagents	3163.8-151.6	1251.7	305
5	TacsMan	3163.8-180	1065.7	315
6	CaiserSose	3163.8-294	1219.7	315
7	polimi_bot	3163.8-306.2	980.8	316
8	umbctac	3163.8-399	1288.4	313

(b) TAC-01 final round.

Rank	Agent	Avg	Std Dev	Games
1	livingagents	3530.6+139.4	622.3	24
2	ATTac	3530.6+91	691.6	24
3	SouthamptonTAC	3530.6	568.8	23
4	whitebear	3530.6-17.4	700.1	24
5	Urlaub01	3530.6-109.4	698.3	24
6	Retsina	3530.6-178.8	668.2	24
7	CaiserSose	3530.6-456.5	656.2	24
8	TacsMan	3530.6-671.3	1054.3	24

TABLE XI
RESULT OF TAC-02

(a) TAC-02 seeding round.

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

(b) TAC-02 final round.

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

When predicting the closing price of a hotel auction, three cases are considered[21]: i) when the good and bad hotels are open; ii) when the counterpart hotel auction had just closed (within the previous minute); and iii) when it had been closed for a longer period of time. The corresponding rule bases are shown in Tables V–VII.

In these tables, the hotel ask prices (P and CP) are expressed in the fuzzy linguistic terms: *very-high*, *high*, *medium*, and *low* [Fig. 6(a) and Table VIII] and the price changes (C and CC) in the fuzzy linguistic terms *quick*, *medium*, and *slow* [Fig. 6(b) and Table IX]. The output of the rule base is the prediction Δ' of how much the price of the given hotel is likely to increase, thus $\Delta' \in \{\text{small, medium, big, very-}big\}$ is the increase that is added to the current price to obtain the predicted clearing price.

2) Hotel Bid Withdrawal: Turning now to the notion of withdrawing bids. TAC does not allow hotel bid withdrawal during a game (as described in Section II). Nevertheless, our agent can effectively achieve withdrawal by the following means. The agent decides which bids to continue with, which bids to withdraw and where new bids are needed based on the output of the Allocator and Hotel Bid Adjustor. The agent decides to withdraw a bid either because the hotel 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 and our agent has already placed a bid higher than a. The agent calculates the predicted clearing price of each hotel auction [from (8)]. The idea is that the agent submits 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 does not violate the rules of the TAC auction, but avoids getting high price hotels. This method proved very effective and meant our agent could withdraw bids before the ask price rose too high. This ability to withdraw bids make it possible to consider switching the bidding between the different hotel types (to be discussed in Section III-G-III) and so increased the flexibility of our bidding strategy.

3) Switching Between Hotels: SouthamptonTACs initial allocation of hotel rooms starts early in the game, while the ask prices of all the hotels are very low. At this time, it uses the reference hotel prices as input to the Allocator. This reference price comes from the average prices of the various hotels in previous ten games. During the course of a game, it is sometimes useful for an agent to change from trying to buy one sort of hotel to going for another. However there are risks in this: i) it may fail to get rid of its existing bids for its original hotels (thus it may double bid); and ii) it is possible that the price of the hotel that is changed to rises very quickly while the price of the old type remains unchanged (thus, the agent may want to switch its bidding back to its original hotel).

To manage the process of determining when to change the type of hotel to bid for, our agent employs the following process. The output of the Allocator is an optimized solution which means that if a new allocation (involving a different hotel) produces one more unit of profit than the current one, it will be suggested. However, blindly following this recommendation may cause the agent to oscillate in its behavior and lead it to having unwanted hotel rooms at the end of the process. To avoid this, our agent makes sure that any change in behavior is likely to have a worthwhile effect on its score. In more detail, given a change threshold θ, suppose a customer is currently allocated

[21]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.

TABLE XII
EXPERIMENT SET-UP FOR CONTROLLED EXPERIMENTS. THE LIGHT GREY AREA INDICATES COMPETITIVE ENVIRONMENTS AND DARK GREY NON-COMPETITIVE ONES

	Number of RA–agents							
Number of RS–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

to stay in Hotel A giving it a utility of U_A. Now assume the Allocator suggests placing this customer in Hotel B giving a utility of U_B (where $U_B > U_A$). The rule for enacting this decision is: **IF** $U_B - U_A - q_A \times p_A > \theta$ **THEN** *change to* B **ELSE** *stay in* A, where $q_A \times p_A$ indicates the loss if the agent cannot withdraw its existing bids in hotel auction A. Here, q_A is the quote price of auction A and p_A is the possibility of not withdrawing the bid (this can be approximated from the speed with which bids have changed in the last minute). Adhering to this rule means we only withdraw those bids that have a continuously changing bid history in the past minute and those where much more profit can be obtained by changing the hotel type.

H. 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 (see Section IV for more details). 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 [21] dynamically varies its bidding strategy according to its assessment of the environment type (see Section IV-D 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 (as discussed in Section III-E). 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.

IV. EVALUATION

Our evaluation of SouthamptonTAC is composed of two components: i) the results from the TAC-01 and TAC-02 competitions, and ii) our *post-hoc* systematic analysis in a range of controlled environments.

A. TAC Results

TAC consisted of a preliminary round (mainly used for practice and fine tuning), a seeding round, the semi-finals and the final round. The seeding round determined groupings for the semi-finals. The top 16 agents were organized into two "heats" for the semi-finals based on their position in the seeding round and the first four teams in both heats entered into the final round.

In TAC-01, SouthamptonTAC-01 obtained the highest score in the seeding round [see Table X(a)]. Table X(b) shows the result of the final round, here SouthamptonTAC-01 had the 3rd highest score.[22] Overall, during the course of the competition some 600 games were played and SouthamptonTAC-01 had the highest mean score and lowest standard deviation.

For TAC-02, Table XI(a) shows the seeding round 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. Table XI(b) 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% (when all the games in the competition are considered, SouthamptonTAC is 2.2% better than whitebear).

When both competitions are considered, there are 12 agents that participated in both TACs (some of the agents are designed by the same group but with different agent names). Four of these twelve qualified for the final rounds in both competitions. For these four agents, their average scores over both competitions (some 1200 games) are: SouthamptonTAC (3229), whitebear (3119), livingagents (3016) and Thalis/CaiserSose (2863). Thus, our agent is the most successful of these.

Through both competitions, 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 TAC situations.

B. 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.[23] 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)** Based on the behavior of the livingagents, UMBCTAC, and Walverine agents (see

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

[23]It is interesting to compare agents that use fuzzy sets with agents that use fixed boundaries are the rules. However, we leave it as our future work.

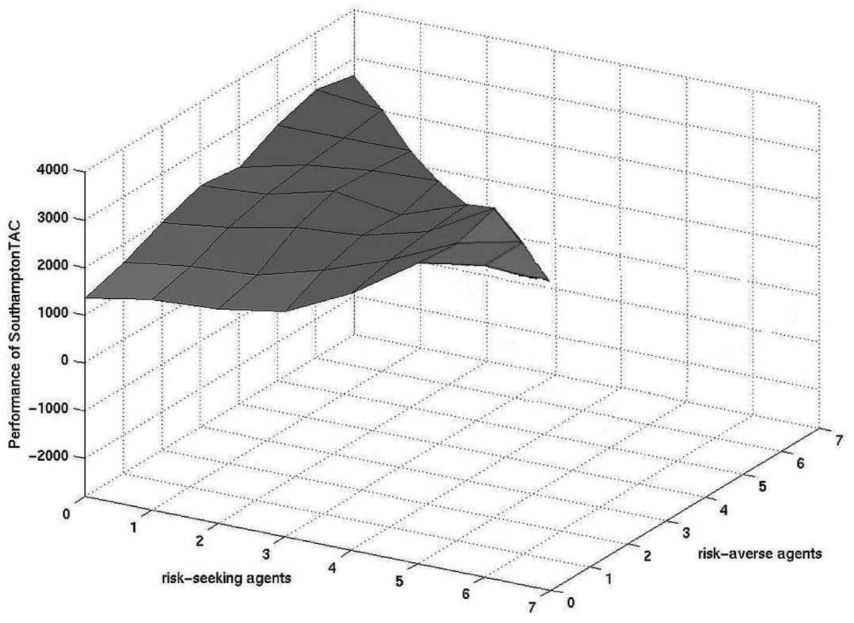


Fig. 7. Performance of SouthamptonTAC in different environments.

Section V 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)** Based on the behavior of the SouthamptonTAC-01, Retsina, and sics agents (see Section V for more details), this agent buys a small number of flight tickets at the beginning of each game to leave some flexibility for changing 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 setup of the experiment is shown in Table XII 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 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 games[24] 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 III-A) it will be and the more RA-agents there are, the less competitive it will be.

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

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

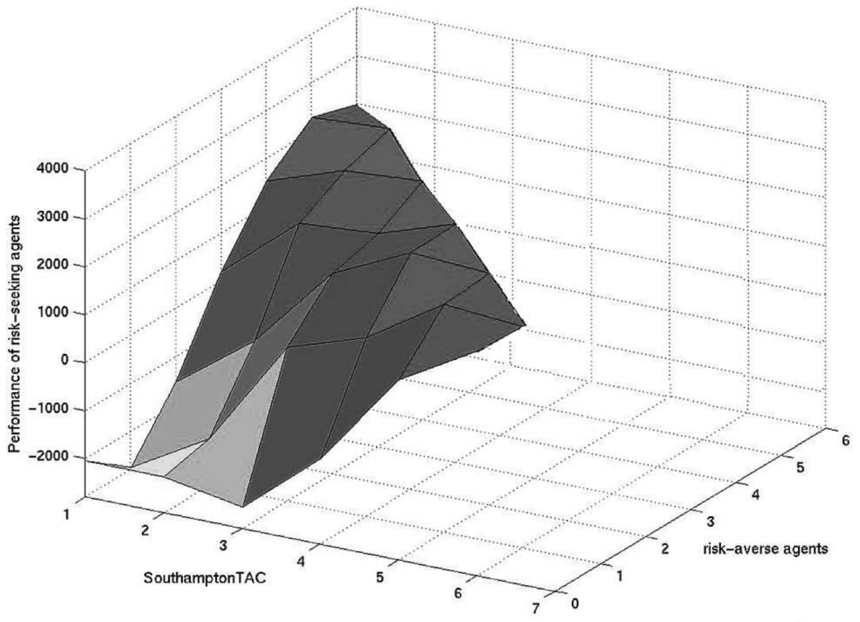
In terms of Conjecture 1, the average hotel clearing price for those environments marked as competitive (in Table XII) is 240 and for those marked as noncompetitive it is 67 and for the remainder it is 125. Thus, Conjecture 1 can be seen to hold.

We now start to analyze the performance in the different environments. Fig. 7 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 agents is $8 - N_{RS} - N_{RA}$. The z-axis shows the average score[25] of SouthamptonTAC. The higher the score, the better the agent performs. Fig. 8 show the performance of the RS-agents and RA-agents on similar graphs.

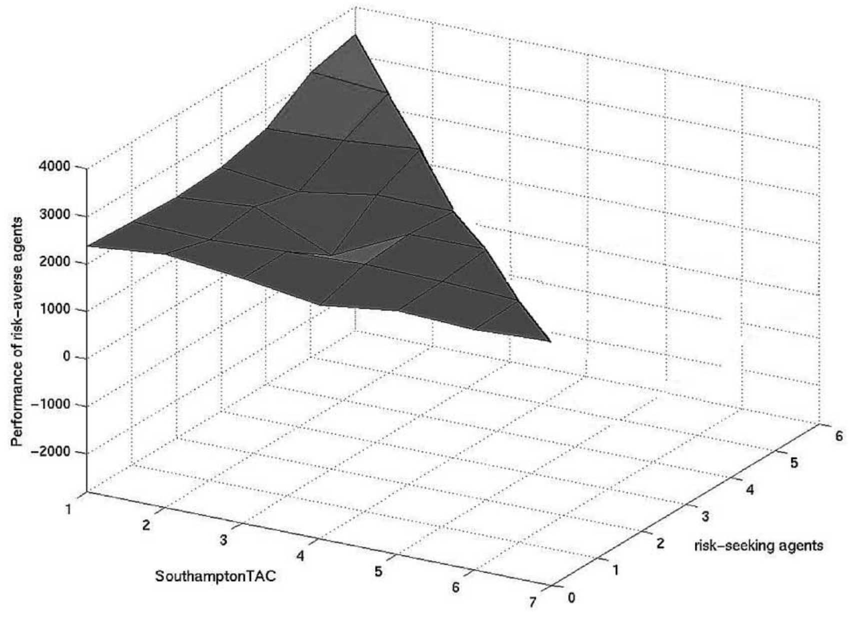
As shown in Fig. 7, SouthamptonTAC does best, obtains the highest score, in competitive games (i.e., where the number of RS-agents is big). This is due to the adaptive nature of its strategy. When finding the game competitive, it alters its strategy in the direction of being risk-averse. In noncompetitive environments, it also does well since it adapts its strategy to bid aggressively because it can always obtain the 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 behavior 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.[26] For RS-agents, as shown in Fig. 8(a), the results also support Conjecture 2. RS-agents behave very well in noncompetitive

[25]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 $\sum_{i=1}^{m} s_i/m$.

[26]The agent that can adapt itself best to the change of environment will survive if there are other agents use similar strategies. Thus, the online parameter adaptation is important for the agent (more discussion can be found in Section VI).



(a) Performance of Risk-seeking agents.



(b) Performance of Risk-averse agents.

Fig. 8. Performance of different risk attitude agents in different environments.

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 semi-competitive games when there are a few RS-agents and SouthamptonTAC agents (see Fig. 8(b)). In the latter two cases, RA-agents change their customers' travel packages reasonably often and this causes them to buy extra hotels and flights that they cannot subsequently use.

Moreover, from Figs. 7 and 8, 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 Figs. 7 and 8 show the performance of a single type of agent in various environments, Fig. 9 compares their scores. There are 8 subfigures and each of them represents several cases of the aforementioned experiments.[27] We found that when the number of SouthamptonTAC agents is small

[27]For example, in figure (c), there are 2 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.

(less than 4), they can always outperform both RS-agents and RA-agents [as shown in (e)–(h) and some cases in (a)–(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 behavior 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).

C. Predicting Hotel Prices

Most of the agents in TAC engage in some form of hotel price prediction (see Section V). Since, generally speaking, the more accurately the agent can predict these prices the more easily it can identify profitable actions. To this end, Table XIII shows the accuracy of SouthamptonTAC's predictions on a minute by 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 XIV 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[28] and the minimum is 8. These results are consistent with those of Table XIII and show that the later a hotel closes, the more accurate our agent's prediction is.

D. 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.[29]

V. Related Work

There are a number of strands of work that are related to what we have described in this paper. First, there is the work on agents bidding in multiple simultaneous auctions. Second, the work using fuzzy techniques to manage an agent's interactions. Third, alternatives to fuzzy reasoning for coping with the uncertainties in bidding. Finally, other agents developed for the TAC. Each of these is now dealt with in turn.

First, we consider bidding in multiple simultaneous auctions. Preist designed an algorithm for agents that participate in multiple English auctions [39]. The algorithm proposes a co-ordination mechanism that can be used in cases where all the auctions terminate simultaneously, and a learning method to tackle auctions that are terminating at different times. However, their strategy is targeted at a single auction protocol (English) and the goods are not inter-dependent. Byde also describes a dynamic programming approach for agents that participate in multiple English auctions to buy a single item [5]. Moreover, in [4], the dynamic programming approach is compared with other algorithms in order to determine its quality. As a result, the dynamic programming approach is shown to be effective. However this method is only for one kind of auction (English) and it only deals with purchasing one item. We also believe it is difficult to extend this approach to a time constrained environment, such as the TAC, because of the heavy computational demands of this technique.

Byde *et al.* framework [6] that enables an agent to make rational decisions across multiple heterogeneous auctions (English, Dutch, First-price sealed-bid, and Vickrey auctions). It does this by using a fixed-auction strategy and a fixed threshold strategy to estimate the expected utility of a bid, and then exploiting a heuristic algorithm to approximate this decision making behavior. However again there is no notion of purchasing inter-related goods.

Anthony *et al.* [1] also propose an approach for agents to bid for a single item in English, Dutch, and Vickrey auctions. The agent decides what to bid based on four parameters: i) the remaining time; ii) the number of remaining auctions; iii) the desire for bargain; and iv) the desperateness of the agent. The overall strategy is to combine these four tactics using a set of relative weights provided by the user. The agent also has a deadline for obtaining the good, but only one item is purchased. In an extension to this model [2], a genetic algorithm is used to search the effective strategies so that an agent can behave appropriately according to its assessment of its prevailing circumstances. Nevertheless, it still does not deal with inter-related goods.

[28]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).

[29]A t-test showed that the adaptive agent's performance is significantly better than the nonadaptive one, where $p < 0.05$.

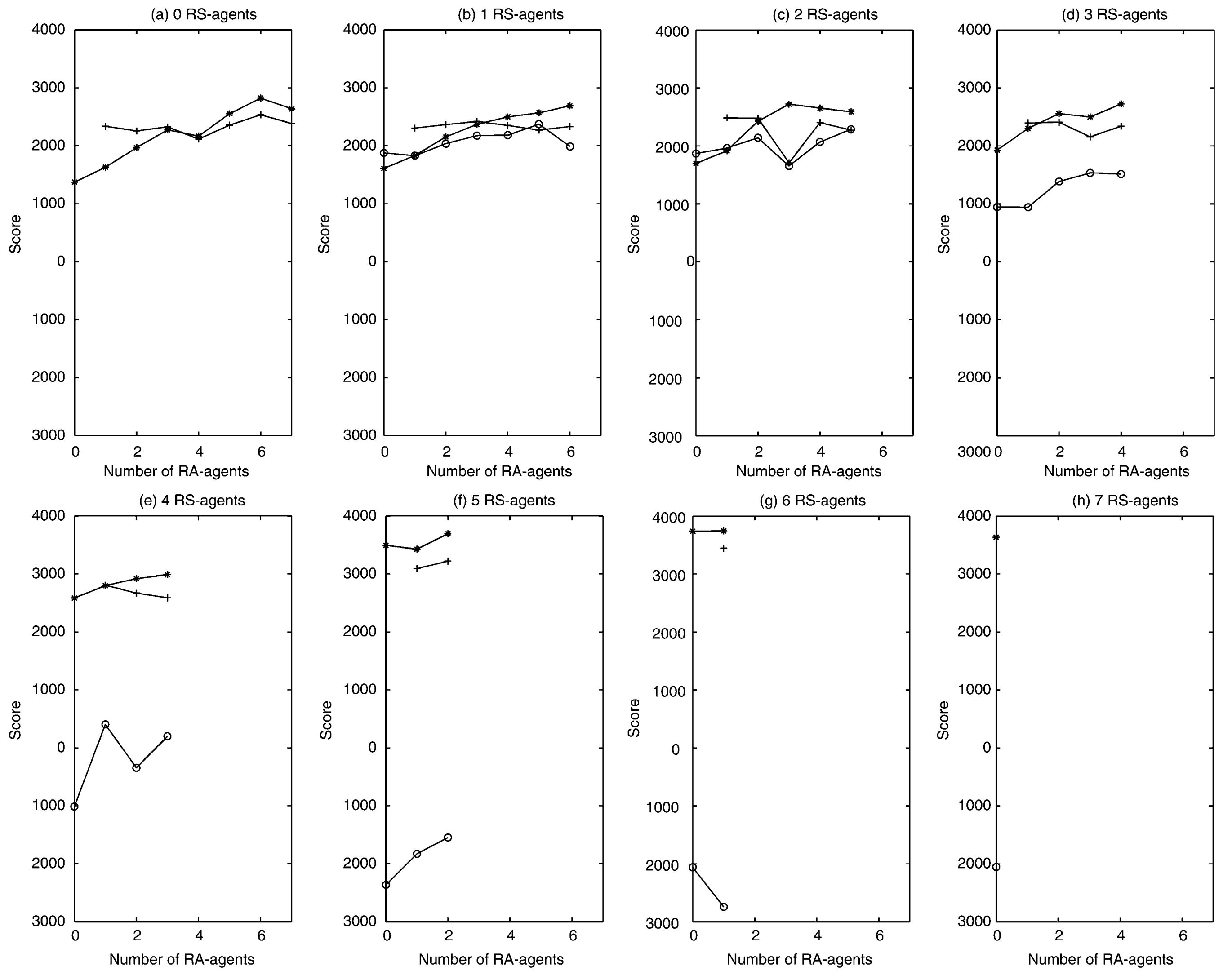


Fig. 9. Relative performance of the agents in different environments. SouthamptonTAC: —*—. RA-agent: —+—. RS-agent: —o—.

Combinatorial auctions do deal with inter-related goods in that they allow bidders to bid for combinations of items. In contrast with the multiple auctions scenarios above, however, combinatrial auctions place the complexity on dealing with the inter-related aspect of the bidding on the auctioneer rather than on the bidding agent. Thus the auctioneer needs a winner determination algorithm to select a set of nonconflicting bids that maximize its revenue. This problem has been shown to be NP-complete [15] and accordingly, a number of algorithms have been developed to achieve this according to various criteria (e.g., anytime solutions [15], polynomial solutions [8], and optimal solutions [40]).

We now turn to the use of fuzzy techniques to manage an agent's interactions. Faratin *et al.* [10] used fuzzy similarity to compute tradeoffs among multiple attributes during bilateral negotiations. An agent first generates some (all) of the potential contracts for which it receives a score θ. Then, the agent finds the contract on the indifference curve for θ which has the maximum similarity degree to the last proposal from the negotiation opponent. Here, fuzzy techniques are used to deal with a bilateral negotiation and the algorithm aims to find a

TABLE XIII
ACTUAL VERSUS 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 XIV
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

win–win (cooperative) solution for both parties. Kowalczyk and Bui [27], [28] modeled the multi-issue negotiation process as a fuzzy constraint satisfaction problem (FCSP). Their approach performs negotiation on individual solutions one at a time. During negotiation, an agent evaluates the offers, relaxes the preferences and constraints and makes counter-offers to find an agreement for both parties. The issues negotiated over actually correspond to the constrained variables and the preferences, constraints and each party's objectives are expressed as fuzzy constraints over these issues. Using this method, the FCSP is to find a solution that maximizes the satisfaction of all constraints of the parties. Luo *et al.* [29], [30] developed a fuzzy constraint based framework for bilateral multi-issue negotiations in semi-competitive trading environments. The framework is expressed via two knowledge models, one for the seller agent and one for the buyer agent. The seller agent's domain knowledge consists of its multi-dimensional representation of the products or services it offers. The buyer agent's domain knowledge consists of the buyer's requirement/preference model (a prioritised fuzzy constraint problem) and buyer's profile model (fuzzy truth propositions). The buyer and seller agents exchange offers and counter-offers with additional constraints revealed or existing constraints being relaxed. Finally, a solution is found if there is one.

We also consider the alternatives to fuzzy reasoning for handling uncertainty in agent interactions (see [32] for a comprehensive survey about handling uncertainty in agent systems). As stated, we chose fuzzy logic-based methods because they have proven to be a practicable solution in solving decision making problems under uncertainty. Fuzzy rules are the most visible manifestation of this approach and have been successfully used in industrial applications, manufacturing, process control, automotive control, and financial trading [54]. There are, however, alternative techniques for handling uncertainty. For example, the possibility-based approach [16], [34] has been used to perform multiagent reasoning under uncertainty for bilateral negotiations. In this paper, uncertainties due to the lack of knowledge about other agents' behaviors are modeled by possibility distributions. Based on information from a case base of previous negotiation behaviors, the possibility distributions are generated by choosing the most similar situation to the current context and the most similar price from the case base. Since this approach relies on a case base, it is unclear what would happen if no similar situations were available. Moreover, even if a similar case exists, it is possible that the strategy used successfully in that situation does not work in the current environment due to the variety of competitors. The Bayesian learning method [55] has also been used to explicitly model multi-issue negotiation in a sequential decision making model. In this paper, a Bayesian network is used to update the knowledge and belief each agent has about the environment and other agents, and offers and counter-offers between agents during bilateral negotiations are generated based on Bayesian probabilities. However, this method is inappropriate in our context because assigning prior probabilities of a bid (ask) being accepted is practically impossible given the dynamism and uncertainty of the multiple auction context.

Finally, we discuss the most successful agents from both competitions in Table XV. *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 [44]. *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* [14] 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* [19] 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 optimizer 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* [7] predicts the hotel closing prices by calculating the Walrasian competitive equilibrium of the game. *whitebear* [47] 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 seen from the aforementioned discussion, there are a number of commonalities between the designs. Firstly, a variety of AI techniques including fuzzy reasoning, machine learning, planning, Markov decision making and heuristic search are used for making predictions about the likely future state of affairs. Thus, 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. Secondly, 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 III-H), 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.

VI. Conclusion and Future Work

This paper presents a successful application of fuzzy set theory in agent-mediated electronic commerce. It details the design, implementation and evaluation of SouthamptonTAC; an agent that successfully participated in both the TAC-01 and the TAC-02 competitions and that employs a range of fuzzy techniques at its core. Specifically, it uses fuzzy pattern recognition to determine the type of environment it is situated in and then uses an adaptive bidding strategy to change its strategy depending on this assessment. In entertainment auctions, the agent continuously observes the current market asks/bids and uses fuzzy set techniques to extend the asks/bids it will accept by decreasing the similarity degree of the fuzzy sets. Moreover, the agent uses fuzzy reasoning techniques (Sugeno controller) to predict the hotel closing prices given the prevailing market conditions.

SouthamptonTAC has been shown to be successful across a wide range of TAC environments (in both competitions as well as in our controlled experiments). 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, online trading environments. These attributes include a time constrained environment, network latency, unpredictable opponents, multiple heterogeneous auction types, and the need to purchase inter-related goods. Given this, we believe that a number of technologies and insights from our work are applicable in a broader agent-mediated e-commerce context and our future work aims to exploit these.

We start by considering the reusable technologies from our work. First, our fuzzy reasoning methods for predicting hotel closing prices can be reused in multiple auction applications in which an agent tries to procure a good from a number inter-related auctions that are running concurrently. In such contexts, the task for a bidder is to select one or more auctions in which to bid. This procedure is usually based on a utility analysis so that the auctions selected will maximize the utility of the bidder (as in [6] discussed in Section V). However, in order to calculate the utility, the likely clearing price of each auction must be estimated first (which is where our fuzzy reasoning techniques can be used). Due to the uncertainty of the context, it is difficult to use a probability-based method to calculate the correlation between auctions because assigning prior probabilities is often impossible in practice in this context. However, our fuzzy reasoning is intuitive and its embodiment in fuzzy rules means that it should be readily comprehensible to the agent's designers (as has happened in other similar applications of fuzzy rules [24], [43], [52]). For example, when faced with multiple inter-related auctions, the following factors can be used in the prediction rules: the auction closing time, the goods purchased, the current ask price of a given auction, the current winning bid and so on. Secondly, our method for an agent assessing its environment is applicable in more general settings. We believe that when agents have to use heuristic strategies it is likely that no one heuristic is likely to be best for all cases. Therefore, such agents need the ability to tailor their strategy according to their assessment of the prevailing situation. To do this they need to be able to determine what environment they are in so that they can best respond. For example, in the general case of multiple auctions, the environment sensor can work based on the previous clearing prices, the type of participants in the auctions, the number of participants in the auctions and so on. Thirdly, the strategy used in entertainment bidding is applicable to any other type of CDA. In a CDA where an agent submits both asks and bids, most agents calculate an optimal bid price that can maximize its utility, and wait for the bid to be matched by other agents (e.g., [17]). However, the strategy employed by our agent uses the similarity in fuzzy sets which allows a greater degree of flexibility in that it will accept bids and asks if they are close enough to the optimal bid.

TABLE XV
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

Now, we turn to the insights from this work that are more broadly applicable. First, we believe that the uncertainty that is inherent in most complex auction 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 behavior 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. Thus, our agent was tuned between rounds based on past performance and the risk attitude of its opponents in the competition. This was possible because the opponents were known in advance for the semifinals and finals and because the opponents' behaviors can be studied in previous rounds. Second, the fuzzy technologies we developed for making predictions, assessing the prevailing context and adapting behavior are computationally efficient and are able to operate with relatively 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. Fourth, while an agent should certainly be responsive to its prevailing context; it should not respond to each and every minor perturbation in the environment. For example, in hotel bidding, the current ask prices of each auction change on a minute by minute basis. Now, it may be that the agent believes it can obtain an improvement in utility by switching its customers between the good and the bad hotels. However, if this improvement is only small then the agent should not switch because its estimation is based on uncertain predictions and if these predictions are slightly out then there may not be a real improvement. Moreover, by making such a switch the agent is taking a risk because it may not be able to off-load those hotels that it has already bought and so it may have to pay for hotels that it cannot use.

For the future, there are some extensions for this model. First, the pattern recognition procedure adopted to classify the environment competitive degree could be enhanced by integrating the decision process with further variables representative of the trading evolution. This is aimed to develop a more robust knowledge base. Second, the membership function parameters need to be adapted dynamically in order to address the inherent time varying nature of trading applications. Here, fuzzy neural network techniques seem appropriate.

APPENDIX

A. *Setup for Allocator-1*

- Notation: i) Let the 20 travel packages be described in the following way: (*outdate indate hotel-type*), where *outdate* $\in \{1,\ldots,4\}$ indicating date out to Tampa; *indate* $\in \{2,\ldots,5\}$ indicating the date back to TACtown; *hotel-type* $\in \{0,1\}$ where 0 stands for S hotel and 1 for T hotel. The 20 valid travel packages can be expressed as: (1 2 1), (1 3 1), (1 4 1), (1 5 1), (2 3 1), (2 4 1),(2 5 1), (3 4 1), (3 5 1), (4 5 1), (1 2 0), (1 3 0), (1 4 0), (1 5 0), (2 3 0), (2 4 0), (2 5 0), (3 4 0), (3 5 0), and (4 5 0). ii) The entertainment tickets have three types for four days. Let (*type day*) denote a ticket, where *type* $\in \{1,2,3\}$ and *day* $\in \{1,2,3,4\}$. For each customer, various tickets can be expressed as: (1 1), (1 2), (1 3), (1 4), (2 1), (2 2), (2 3), (2 4), (3 1), (3 2), (3 3), and (3 4).
- Variables: i) For customer $i \in \{1,\ldots,8\}$, there are 20 variables $f_{i,j} \in \{0,1\}$ where $j \in \{1,\ldots,20\}$, each representing the jth travel package. Customer i is allocated to package j when $f_{i,j} = 1$. Thus, there are 160 variables for eight customers. ii) BUY[0], ..., BUY[3] represent the inflight tickets to buy; BUY[4], ..., BUY[7] represent the outflight tickets to buy; BUY[8], ..., BUY[11] represent the T rooms to bid for; BUY[12], ..., BUY[15] represent the S rooms to bid for. iii) For each customer $i \in \{1,\ldots,8\}$, there are 12 variables $e_{i,j} \in \{0,1\}$ where $j \in \{1,\ldots,12\}$, each representing the jth entertainment ticket. Customer i is allocated ticket j when $e_{i,j} = 1$.
- Constants: Let OWN[0], ..., OWN[3] be the inflight tickets owned by the agent; OWN[4], ..., OWN[7] be the outflight tickets owned by the agent; OWN[8], ..., OWN[11] represent the T rooms owned by the agent; OWN[12], ..., OWN[15] represent the S rooms owned by the agent; and OWN[16], ..., OWN[27] denote each kind of entertainment ticket owned by the agent.
- Constraints
  i) The number of hotel rooms must be less than the number of rooms the agent owns and that it will bid for (eight constraints)

$$\sum_{i=1}^{8}(f_{i,1}+f_{i,2}+f_{i,3}+f_{i,4}) \le \text{OWN}[8]+\text{BUY}[8]$$

$$\sum_{i=1}^{8}(f_{i,2}+f_{i,3}+f_{i,4}+f_{i,5}+f_{i,6}+f_{i,7}) \le \text{OWN}[9]+\text{BUY}[9]$$

$$\sum_{i=1}^{8}(f_{i,3}+f_{i,4}+f_{i,6}+f_{i,7}+f_{i,8}+f_{i,9}) \le \text{OWN}[10]+\text{BUY}[10]$$

$$\sum_{i=1}^{8}(f_{i,4}+f_{i,7}+f_{i,9}+f_{i,10}) \le \text{OWN}[11]+\text{BUY}[11]$$

$$\sum_{i=1}^{8}(f_{i,11}+f_{i,12}+f_{i,13}+f_{i,14}) \le \text{OWN}[12]+\text{BUY}[12]$$

$$\sum_{i=1}^{8}(f_{i,12}+f_{i,13}+f_{i,14}+f_{i,15}+f_{i,16}+f_{i,17}) \le \text{OWN}[13]+\text{BUY}[13]$$

$$\sum_{i=1}^{8}(f_{i,13}+f_{i,14}+f_{i,16}+f_{i,17}+f_{i,18}+f_{i,19}) \le \text{OWN}[14]+\text{BUY}[14]$$

$$\sum_{i=1}^{8}(f_{i,14}+f_{i,17}+f_{i,19}+f_{i,20}) \le \text{OWN}[15]+\text{BUY}[15].$$

  ii) For customer $i \in \{1,\ldots,8\}$ entertainment tickets must be used between the days in Tampa (32 constraints)

$$\begin{aligned} &f_{i,1}+f_{i,2}+f_{i,3}+f_{i,4}+f_{i,11}+f_{i,12} \\ &\quad + f_{i,13}+f_{i,14} \\ &\qquad \ge e_{i,1}+e_{i,5}+e_{i,9} \\ &f_{i,2}+\cdots+f_{i,7}+f_{i,12}+\cdots+f_{i,17} \\ &\qquad \ge e_{i,2}+e_{i,6}+e_{i,10} \\ &f_{i,3}+f_{i,4}+f_{i,6}+\cdots+f_{i,9}+f_{i,13} \\ &\quad + f_{i,14}+f_{i,16}+\cdots+f_{i,19} \\ &\qquad \ge e_{i,3}+e_{i,7}+e_{i,11} \\ &f_{i,4}+f_{i,7}+f_{i,9}+f_{i,10}+f_{i,14} \\ &\quad + f_{i,17}+f_{i,19}+f_{i,20} \\ &\qquad \ge e_{i,4}+e_{i,8}+e_{i,12}. \end{aligned}$$

  iii) Each customer $i \in \{1,\ldots,8\}$ has only one valid package (8 constraints), $\sum_{j=1}^{20} f_{i,j} \le 1$.
  iv) For entertainment ticket $j \in \{1,\ldots,12\}$ the number of tickets used must be less than the number the agent owns (12 constraints):

$$\sum_{i=1}^{8} e_{i,j} \le \text{OWN}[j+15].$$

  v) The flights tickets that can be used must be less than the number the agent owns or that it will buy (eight constraints)

$$\sum_{i=1}^{8}(f_{i,1}+f_{i,2}+f_{i,3}+f_{i,4}+f_{i,11}+f_{i,12}+f_{i,13}+f_{i,14}) \le \text{OWN}[0]+\text{BUY}[0]$$

$$\sum_{i=1}^{8}(f_{i,5}+f_{i,6}+f_{i,7}+f_{i,15}+f_{i,16}+f_{i,17}) \le \text{OWN}[1]+\text{BUY}[1]$$

$$\sum_{i=1}^{8}(f_{i,8}+f_{i,9}+f_{i,18}+f_{i,19}) \leq \mathrm{OWN}[2]+\mathrm{BUY}[2]$$
$$\sum_{i=1}^{8}(f_{i,10}+f_{i,20}) \leq \mathrm{OWN}[3]+\mathrm{BUY}[3]$$
$$\sum_{i=1}^{8}(f_{i,1}+f_{i,11}) \leq \mathrm{OWN}[4]+\mathrm{BUY}[4]$$
$$\sum_{i=1}^{8}(f_{i,2}+f_{i,5}+f_{i,12}+f_{i,15}) \leq \mathrm{OWN}[5]+\mathrm{BUY}[5]$$
$$\sum_{i=1}^{8}(f_{i,3}+f_{i,6}+f_{i,8}+f_{i,13}+f_{i,16}+f_{i,18}) \leq \mathrm{OWN}[6]+\mathrm{BUY}[6]$$
$$\sum_{i=1}^{8}(f_{i,4}+f_{i,7}+f_{i,9}+f_{i,10}+f_{i,14}+f_{i,17}+f_{i,19}+f_{i,20}) \leq \mathrm{OWN}[7]+\mathrm{BUY}[7].$$

vi) For customer $i \in \{1,\ldots,8\}$ each type of entertainment ticket can only be used once (24 constraints):

$$\sum_{j=1}^{4} e_{i,j} \leq 1$$
$$\sum_{j=5}^{8} e_{i,j} \leq 1$$
$$\sum_{j=9}^{12} e_{i,j} \leq 1.$$

- Objective function. The objective function is to maximize the following formula:

$$\sum_{i=1}^{8}\sum_{j=1}^{20}(f_{i,j}\times u_{i,j})+\sum_{i=1}^{8}\sum_{j=1}^{12}(e_{i,j}\times E_{i,j}) - \sum_{p=0}^{15}(\mathrm{BUY}[i]\times \mathrm{Price}[i])$$

where $u_{i,j}$ is the utility if customer i chooses package j; $E_{i,j}$ is the preference value if customer i enjoys entertainment ticket j; and Price$[i]$ is the updated ask price of the corresponding auctions.

B. Setup for Allocator-2

- Variables (276 variables). The difference with *allocator-1* is that there are no variables BUY[8], ..., BUY[15], since all the hotel auctions have closed. However, BUY[16], ..., BUY[27], indicating the number of entertainment tickets to buy for each of the 12 tickets, are added.
- Constraints (92 constraints). All but the following are the same as those of *allocator-1*: Constraints v) are changed to

$$\sum_{i=1}^{8} e_{i,j} \leq \mathrm{OWN}[j+15]+\mathrm{BUY}[j+15]$$

where $j \in \{1,\ldots,12\}$. This means the allocator can also calculate which entertainment tickets to bid for in order to get the optimal utility. All the other constraints are the same as *allocator-1*.
- Objective function. The objective function is to maximize the following formula:

$$\sum_{i=1}^{8}\sum_{j=1}^{20}(f_{i,j}\times u_{i,j})+\sum_{i=1}^{8}\sum_{j=1}^{12}(e_{i,j}\times E_{i,j}) - \sum_{p=0}^{7}(\mathrm{BUY}[i]\times Price[i]) - \sum_{p=16}^{27}(\mathrm{BUY}[i]\times Price[i])$$

where $u_{i,j}$ is the utility if customer i chooses package j; $E_{i,j}$ is the preference value if customer i enjoys entertainment ticket j; and $Price[i]$ $(0 \leq i \leq 7)$ is the updated ask price of the flight auctions and $Price[i](16 \leq i \leq 27)$ is the entertainment ticket price in the corresponding auctions.

Acknowledgment

The authors would like to thank the TAC team at the University of Michigan (TAC-01) and the Swedish Institute of Computer Science (TAC-02), including M. Wellman, K. O'Malley, W. E. Walsh, D. R. C. Kiekintveld, J. Ericsson, and N. Finne for setting up the TAC server and the competition and for responding so promptly to our requests. They would also like to thank a number of people from their department: M. Luck who went to the U.S. to present our agent at TAC-01 on very short notice, Y. Z. Wei who helped with agent connecting problems and monitoring the seeding round games in TAC-02, A. J Riddoch, J. Hallett, and T. Chown who helped with various network difficulties that we experienced in both competitions, and X. Luo for his encouragement and support during the whole course of the competition.

References

[1] P. Anthony and N. R. Jennings, "Evolving bidding strategies for multiple auctions," in *Proc. 15th Eur. Conf. Artificial Intelligence*, F. Van Harmelen, Ed., Amsterdam, The Netherlands, 2002, pp. 178–182.
[2] ——, "Developing a bidding agent for multiple heterogeneous auctions," *ACM Trans. Internet Technol.*, vol. 3, no. 3, pp. 185–217, 2003.
[3] R. Bapna, P. Goes, and A. Gupta, "Insights and analyzes of online auctions," *Commun. ACM*, vol. 44, no. 11, pp. 42–50, 2001.
[4] A. Byde, "A comparison among bidding algorithms for multiple auctions," Hewlett Packard Research Labs, Bristol, U.K., Tech. Rep. HPL-2001-300, 2001.
[5] ——, "An optimal dynamic programming model for algorithm design in simultaneous auctions," Hewlett Packard Research Labs, Bristol, U.K., Tech. Rep. HPL-2001-67, 2001.
[6] A. Byde, C. Preist, and N. R. Jennings, "Decision procedures for multiple auctions," in *Proc. 1st Int. Joint Conf. Autonomous Agents Multi-Agent Systems*, Bologna, Italy, July 2002, pp. 613–620.
[7] S. Cheng, E. Leung, K. Lochner, K. O'Malley, D. Reeves, L. Schvartzman, and M. Wellman, "Walverine: A walrasian trading agent," in *Proc. 2nd Int. Joint Conf. Autonomous Agents Multi-Agent Systems*, Melbourne, Australia, 2003, pp. 465–472.

[8] V. D. Dang and N. R. Jennings, "Polynomial algorithms for clearing multi-unit single item and multi-unit combinatorial reverse auctions," in *Proc. 15th Eur. Conf. Artificial Intelligence*, F. van Harmelen, Ed., Amsterdam, The Netherlands, 2002, pp. 23–27.

[9] R. Das, J. Hanson, J. Kephart, and G. Tesauro, "Agent-human interactions in the continuous double auction," in *Proc. 17th Int. Joint Conf. Artificial Intelligence*, vol. 2, B. Nebel, Ed., Washington, DC, Aug. 2001, pp. 1169–1176.

[10] P. Faratin, C. Sierra, and N. R. Jennings, "Using similarity criteria to make trade-offs in automated negotiations," *Artif. Intell.*, vol. 142, no. 2, pp. 205–237, 2002.

[11] R. Fisher and W. Ury, *Getting to Yes: Negotiating an Agreement Without Giving*. New York: Random House, 1981.

[12] T. Fraichard and P. Garnier, "Fuzzy control to drive car-like vehicles," *Robot. Auton. Syst.*, vol. 34, no. 1, pp. 1–22, 2001.

[13] D. Friedman and J. Rust, *The Double Auction Market: Institutions, Theories and Evidence*. Reading, MA: Addison-Wesley, 1992.

[14] C. Fritschi and K. Dorer, "Agent-oriented software engineering for successful TAC participation," in *Proc. 1st Int. Joint Conf. Autonomous Agents Multi-Agent Systems*, Bologna, Italy, 2002, pp. 45–46.

[15] Y. Fujishima, K. Leyton-Brown, and Y. Shoham, "Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches," in *Proc. 16th Int. Joint Conf. Artificial Intelligence*, 1999, pp. 548–553.

[16] E. Giménez-Funes *et al.*, "Designing bidding strategies for trading agents in electronic auctions," in *Proc. 3rd Int. Conf. Multi-Agent Systems*, 1998, pp. 136–143.

[17] S. Gjerstad and J. Dickhaut, "Price formation in double auction," *Games Econ. Behavior*, vol. 22, pp. 1–29, 1998.

[18] A. Greenwald, "The 2002 trading agent competition: An overview of agent designs," Brown University, Providence, RI, Tech. Rep., 2002.

[19] A. Greenwald and J. Boyan, "Bidding algorithms for simultaneous auctions: A case study," in *Proc. 3rd ACM Conf. Electronic Commerce*, 2001, pp. 115–124.

[20] R. H. Guttman, A. G. Moukas, and P. Maes, "Agent-mediated electronic commerce: A survey," *Knowledge Eng. Rev.*, vol. 13, no. 2, pp. 147–159, 1998.

[21] M. He and N. R. Jennings, "SouthamptonTAC: An adaptive autonomous trading agent," *ACM Trans. Internet Technol.*, vol. 3, no. 3, pp. 218–235, 2003.

[22] M. He, N. R. Jennings, and H. F. Leung, "On agent-mediated electronic commerce," *IEEE Trans. Knowledge Data Eng.*, vol. 15, pp. 985–1003, July 2003.

[23] M. He and N. R. Jennings, "SouthamptonTAC: Designing a successful trading agent," in *Proc. 15th Eur. Conf. Artificial Intelligence*, F. van Harmelen, Ed., Amsterdam, The Netherlands, 2002, pp. 8–12.

[24] M. He, H. F. Leung, and N. R. Jennings, "A fuzzy logic based bidding strategy for autonomous agents in continuous double auctions," *IEEE Trans. Knowledge Data Engineering*, vol. 15, pp. 1345–1363, Nov. 2003.

[25] N. R. Jennings, "An agent-based approach for building complex software systems," *Commun. ACM*, vol. 44, no. 4, pp. 35–41, 2001.

[26] N. R. Jennings *et al.*, "Automated negotiation: Prospects, methods and challenges," *Int. J. Group Decision and Negotiation*, vol. 10, no. 2, pp. 199–215, 2001.

[27] R. Kowalczyk, "On negotiation as a distributed fuzzy constraint satisfaction problem," in *Proc. DEXA e-Negotiation Workshop*, 2000, pp. 631–637.

[28] R. Kowalczyk and V. Bui, "On fuzzy e-negotiation agents: Autonomous negotiation with incomplete and imprecise information," in *Proc. DEXA e-Negotiation Workshop*, 2000.

[29] X. Luo, "A study of information sharing of heterogenous reasoning models in multi-agent enviornment," Ph.D. dissertation, Univ. New England, Armidale, NSW, Australia, 1998.

[30] X. Luo, N. R. Jennings, N. Shadbolt, H. F. Leung, and J. H. M. Lee, "A fuzzy constraint based model for bilateral, multi-issue negotiation in semi-competitive environments," *Artif. Intell.*, vol. 148, no. 1–2, pp. 53–102, 2003.

[31] X. Luo, C. Zhang, and N. R. Jennings, "A hybrid model for sharing information between fuzzy, uncertain and default reasoning models in multi-agent systems," *Int. J. Uncertainty, Fuzziness Knowledge-Based Syst.*, vol. 10, no. 4, pp. 401–450, 2002.

[32] X. Luo, C. Zhang, and H. F. Leung, "Information sharing between heterogeneous uncertain reasoning models in a multi-agent environment: A case study. newblock," *Int. J. Approx. Reason.*, vol. 27, no. 1, pp. 27–59, 2001.

[33] M. Ma, "Agents in e-commerce," *Commun. ACM*, vol. 42, no. 3, pp. 79–80, Mar. 1999.

[34] N. Matos and C. Sierra, "Evolutionary computing and negotiating agents," in *Agent Mediated Electronic Commerce*. New York: Springer-Verlag, 1998, vol. 1571, Lecture Notes in Artificial Intelligence, pp. 126–150.

[35] S. Mohammadi, I. Hassanzadeh, R. Mathur, and K. Patil, "A new fuzzy decision-making procedure applied to emergency electric power distribution scheduling," *Eng. Applicat. Artif. Intell.*, vol. 13, no. 6, pp. 731–740, 2000.

[36] D. C. Parkes, L. H. Ungar, and D. P. Foster, "Accounting for cognitive costs in online auction design," in *Agent Mediated Electronic Commerce*, P. Noriega and C. Sierra, Eds. New York: Springer-Verlag, 1998, vol. 1571, Lecture Notes in Artificial Intelligence, pp. 25–40.

[37] W. Pedrycz, "Fuzzy sets in pattern recognition methodology and methods," *Pattern Recogn.*, vol. 23, no. 1, pp. 121–146, 1990.

[38] J. M. Perloff, *Microeconomics*. Reading, MA: Addison Wesley, 1998, ch. 2, pp. 29–32.

[39] C. Preist, A. Byde, and C. Bartolini, "Economic dynamics of agents in multiple autions," in *Proc. 5th Int. Conf. Autonomous Agents*, Montreal, QC, Canada, 2001, pp. 545–551.

[40] T. Sandholm, "Algorithm for optimal winner determination in combinatorial auctions," *Artif. Intell.*, vol. 135, pp. 1–54, 2002.

[41] Y. Sawaragi, H. Nakayama, and Tetsuyotanino, *Theory of Multiobjective Optimization*. New York: Elsevier Science, 1985.

[42] C. Sierra and F. Dignum, "Agent-mediated electronic commerce: Scientific and technological roadmap," in *Agent Mediated Electronic Commerce*, F. Dignum and C. Sierra, Eds. New York: Springer-Verlag, 2001, vol. 1991, Lecture Notes in Computer Science, pp. 1–18.

[43] Z. A. Sosnowski, "Comments on fuzzyshell: A large-scale expert systems shell using fuzzy logic for uncertainty reasoning," *IEEE Trans. Fuzzy Syst.*, vol. 8, pp. 817–820, Dec. 2000.

[44] P. Stone, M. L. Littman, S. Singh, and M. Kearns, "Attac-2000: An adaptive autonomous bidding agent," *J. Artif. Intell. Res.*, vol. 15, pp. 189–206, 2001.

[45] M. Sugeno, "An introductory survey of fuzzy control," *Inform. Sci.*, vol. 36, pp. 59–83, 1985.

[46] K. K. Tan and K. Z. Tang, "Vehicle dispatching system based on taguchi-tuned fuzzy rules," *Eur. J. Oper. Res.*, vol. 128, no. 3, pp. 545–557, 2001.

[47] I. Vetsikas and B. Selman, "A principled study of the design tradeoffs for autonomous trading agents," in *Proc. 2nd Int. Joint Conf. Autonomous Agents Multi-Agent Systems*, Melbourne, Australia, 2003, pp. 473–480.

[48] W. Vickrey, "Counterspeculation, auctions and competitive sealed tenders," *J. Finance*, vol. 16, pp. 8–37, 1961.

[49] M. P. Wellman, A. Greenwald, P. Stone, and P. R. Wurman, "The 2001 trading agent competition," in *Proc. 14th Conf. Innovative Applications Artificial Intelligence*, Edmonton, AB, Canada, 2002.

[50] P. R. Wurman, "Dynamic pricing in the virtual marketplace," *IEEE Internet Comput.*, vol. 5, pp. 36–42, Mar./Apr. 2001.

[51] P. R. Wurman, M. Wellman, and W. Walsh, "A parametrization of the auction design space," *Games Econ. Behavior*, vol. 35, pp. 304–338, 2001.

[52] Y. Yam and L. T. Koczy, "Representing membership functions as points in high-dimensional spaces for fuzzy interpolation and extrapolation," *IEEE Trans. Fuzzy Syst.*, vol. 8, pp. 761–772, Dec. 2000.

[53] J. F. F. Yao and J. S. Yao, "Fuzzy decision making for medical diagnosis based on fuzzy number and compositional rule of inference," *Fuzzy Sets Syst.*, vol. 120, no. 2, pp. 351–366, 2001.

[54] J. Yen, "Fuzzy logic—A modern perspective," *IEEE Trans. Knowledge Data Eng.*, vol. 11, pp. 153–165, Jan. 1999.

[55] D. Zeng and K. Sycara, "Bayesian learning in negotiation," *Int. J. Human–Comput. Stud.*, vol. 48, pp. 125–141, 1998.

[56] H.-J. Zimmermann, *Fuzzy Set Theory and Its Applications*. Norwell, MA: Kluwer, 1996, ch. 11, pp. 203–240.

Minghua He received the M.S. degree in computer science from Southwest China Normal University, Chongqing, China, in 1999. She is currently working toward the Ph.D. degree with the School of Electronics and Computer Science at the University of Southampton, Southampton, U.K.

Her research interests center around the study of artificial intelligence, especially the area of agent-based automated auctions for E-commerce.

Nicholas R. Jennings received the B.Sc. (Hons.) degree in computer science from Exeter University, Exeter, U.K., and the Ph.D. degree in artificial intelligence from the University of London (Queen Mary College), London, U.K.

He is a Professor of computer science in the School of Electronics and Computer Science, Southampton University, Southampton, U.K., where he carries out basic and applied research in agent-based computing. He is Deputy Head of School (Research), Head of the Intelligence, Agents, Multimedia Group, and is also the Chief Scientific Officer for lost wax. He helped pioneer the use of agent-based techniques for real-world applications and has also made theoretical contributions in the areas of automated negotiation and auctions, cooperative problem solving, and agent-oriented software engineering. He has published some 200 articles on various facets of agent-based computing, and is in the top 125 most cited computer scientists (according to the citeseer digital library).

Prof. Jennings has won several awards for his research, including the Computers and Thought Award (1999), an IEE Achievement Medal (2000), and the ACM Autonomous Agents Research Award (2003).