An Adaptive Bidding Agent for Multiple English Auctions: A Neuro-Fuzzy Approach

Minghua He, Nicholas R. Jennings and Adam Prügel-Bennett
School of Electronics and Computer Science
The University of Southampton, Southampton, SO17 1BJ, United Kingdom.
E-mail: {mh00r,nrj,apb}@ecs.soton.ac.uk

***Abstract*— This paper presents the design, implementation and evaluation of a novel bidding strategy for obtaining goods in multiple overlapping English auctions. The strategy uses fuzzy sets to express trade-offs between multi-attribute goods and exploits neuro-fuzzy techniques to predict the expected closing prices of the auctions and to adapt the agent's bidding strategy to reflect the type of environment in which it is situated. We show, through empirical evaluation against a number of methods proposed in the multiple auction literature, that our strategy performs effectively and robustly in a wide range of scenarios.**

I. Introduction

Online auctions are increasingly being used for a variety of e-commerce applications. This wide-spread adoption means that in many cases there are likely to be multiple auctions selling the desired good or service. Given this huge search space, software agents have an important role to play. They can monitor relevant auctions, compare and make trade-offs between the offerings, and determine what bids to place in the chosen auctions in order to obtain the best deals.

Against this background, this paper develops a bidding strategy for bidding across multiple auctions. We specifically consider the case of bidding in multiple overlapping English auctions. An English auction is one in which a single good is on offer and the auctioneer starts with his reserve price (minimum acceptable) and solicits successively higher (public) bids from the bidders and the last bidder remaining in the auction is the winner [4]. In contrast to the stand-alone English auction, there is no dominant strategy that can be exploited in the multiple auction context.

In more detail, we consider a multiple English auction market, where each auction is given a start and an end time that may overlap with other auctions. Each auction sells a single unit of the desired good and this good may be described by multiple attributes (*e.g.*, in auctions selling flights, the goods may be described by their dates of departure and return, and the class of ticket being bought). Bids for these goods must be at least $h(a)$ larger than the previous price to be valid. If an agent bids successfully, it becomes the active agent which is holding the bid. Auctions respond to any bid before they close and their good is allocated to the active bid holder when the auction closes. Our auctions have a soft deadline, *i.e.*, they do not close until a fixed period after the last bid is placed (e.g., Yahoo!Auctions and Auction Universe). This means sniping is not effective and the lack of deadline effects means the auctions are akin to the standard English one.

Given this context, we aim at developing a bidding agent that buys a single unit of the good from the available set of auctions. As the goods are composed of multiple attributes, the agent may have to make trade-offs between them in its bidding in order to satisfy the user's preferences that include: (i) a valuation v for the good (expressed as a fuzzy set); (ii) the ratings for different values of the good's attributes (expressed as fuzzy sets); and (iii) the weights which balance the valuation and the other attributes.

To cope with the uncertainty inherent in the multi-auction context and to make trade-offs between the different variants of the goods available is a complex decision making problem. A key component is being able to make predictions about the likely closing prices of the various auctions so that the agent can determine whether it should place a bid at the current moment or whether it should delay because better deals may subsequently become available. In our previous work, we successfully used adaptive fuzzy inference methods for this task in continuous double auctions [5] and the Trading Agent Competition [3]. However, in both cases, the parameter adaptation of the fuzzy rules was limited. To rectify this, here we exploit *fuzzy neural networks* (FNNs) [6] since these can do the fuzzy reasoning and, through learning, can adjust the parameters of the fuzzy terms and consequent output as the auctions progress. This adaptation enables the agent's bidding behaviour to better reflect the current state of its environment.

Our work advances the state of the art in the following ways. First, we develop an agent bidding strategy for obtaining goods in multiple overlapping English auctions. A practical algorithm is described and through empirical evaluation it is shown to be effective in a wide range of situations. The strategy adapts to its prevailing circumstances through a mixture of off-line and on-line learning. Second, by exploiting a fuzzy set representation of the user's preferences, the strategy is able to make trade-offs between the various attributes of the goods it purchases and cope with the inherent imprecision/flexibility that often characterises a user's preferences.

The rest of this paper is structured as follows. Section II describes the bidding algorithm and how the FNN operates. Section III gives an example of the FNN strategy in operation in a flight auction scenario. Section IV provides a systematic empirical evaluation of the strategy and benchmarks it against a number of strategies that have been proposed in the literature. Finally, Section V concludes.

II. THE FNN BIDDING STRATEGY

This section details the FNN bidding strategy.

A. The FNN Bidding Algorithm

The agent considers bidding if and only if it is not holding an active bid in an auction or it has not already obtained the good. Assuming neither of these conditions hold, the agent must decide which of the available auctions it should bid in. To do so, it first determines the auction that best satisfies the user's preferences (calculation detailed in section II-D) given its expectation about the closing prices (calculation detailed in section II-B) of each auction. Then, rather than placing a bid in the selected auction immediately, it uses the intervening time before this auction closes to see if it can bid in auctions that close earlier and have an evaluation "close" (a fuzzy term) to that of the selected auction. The intuition here is that given the significant degrees of uncertainty that exist, precise calculations are simply not reliable and an auction that appears slightly less promising may well turn out to be better. Given this, the agent should consider bidding in auctions that have broadly similar expected returns so as to increase its chances of obtaining the item (by participating in more auctions), while ensuring the likely return is one of the highest. Thus, if there are such close auctions, the agent will bid in them in order of increasing closing time (*i.e.*, bid first in the one that is going to close first, then in the one that will close next, and so on). The degree of closeness that is required to trigger bidding is captured by the parameter λ. Then if the difference is within λ, the agent will bid in the auction. In this sense, the choice of λ represents the risk attitude of the user [5].

Given the nature of this decision making task, it is important that the various parameters of the FNN fit the prevailing context as accurately as possible. This is achieved via off-line and on-line learning. In the former case, a number of simulated games are used to set the initial parameters of the FNN. After this, the agent can be used in an operational setting to actually purchase goods. In the latter case, the agent keeps track of the various auctions and when changes occur (*e.g.*, when an auction has closed or the ask prices change) they are fed into the FNN as new training samples. These samples are weighted more highly than older ones and so enable the agent to better reflect prevailing circumstances.

In more detail, the decision making algorithm for the FNN is given in figure 1. In this algorithm, $data$ is the database for storing the collected data required for learning in the FNN and new_data is the data generated since the agent last monitored the auctions. The variable $holding$ is $true$ when the agent has an active bid on hold. An explanation of the algorithm's key functions are as follows. $AuctionRunning$ (line 3) returns $true$ if there are still available auctions to bid in, $false$ otherwise. $Update()$ (line 4) returns changes in the auctions since they were last monitored. Such changes include whether the agent is holding an active bid or has obtained the good, the updated ask price of each auction, the transaction price for any auctions that recently closed, and the number of auctions left to bid in. $Trim()$ (line 6) returns a modified set of $data$ in which the oldest information is removed.

PROCEDURE $FNNrun()$

```
 1: data ← {}
 2: holding ← false
 3: while holding = false or AuctionRunning() do
 4:    new_data ← Update()
 5:    data ← {data, new_data}
 6:    data ← Trim()
 7:    FNNtrain(data)
 8:    if Success() then
 9:       Return
10:    end if
11:    if Hold() = false then
12:       [a_best, s_best] ← FNNpredict()
13:       L ← RunningAuctions()
14:       for all a ∈ L do
15:          if (Evaluate(a) ≥ s_best) or (s_best − Evaluate(a) ≤ λ)
             then
16:             Bid(a)
17:             break
18:          end if
19:       end for
20:    end if
21: end while
```

Fig. 1. The FNN bidding algorithm.

$FNNtrain(data)$ (line 7) trains the FNN on the set of $data$ which involves adjusting the parameters to better reflect the prevailing circumstances (see section II-B). $Success()$ (line 8) returns $true$ if a bid has succeeded in winning the good, $false$ otherwise. $Hold()$ (line 11) returns $true$ if the agent has an active bid in an auction, $false$ otherwise. $FNNpredict()$ (line 12) returns the FNN's current prediction about the best auction (a_{best}) to bid in and the degree to which this satisfies the user's preferences (s_{best}) given the predicted prices of all the auctions. To reason about the expected closing price of each auction, the FNN considers a per auction reference price (p_{ref}), the number of auctions left ($n_{auction}$), and the order in which the auctions are due to close ($o_{auction}$). Here the reference price represents a likely value at which the auction will close for that particular variant of the good. It is computed by considering the current price in that auction, the transaction prices of auctions that have previously sold the specified good, and the average transaction price in the history records for the specified good. $RunningAuctions()$ (line 13) returns a list L of all the running auctions in ascending order of end times. $Evaluate(a)$ (line 15) returns the evaluation of the auction a given its current ask price. This evaluation balances both price and the other attributes of the goods in consideration (see section II-D). $Bid(a)$ (line 16) places a bid in auction a.

To realise this algorithm, a fuzzy neural network needs to be set up (section II-B), the FNN's learning algorithm needs to be defined (section II-C), and an evaluation method needs to be provided for ranking the various auctions (section II-D).

B. The FNN Architecture

As noted above, the FNN takes three inputs (p_{ref}, $n_{auction}$, $o_{auction}$) and has one output (the expected auction closing price p_{close}). To realise this we developed a FNN with 5 layers.

TABLE I

THE FNN RULE BASE.

R_1:	If p_{ref} is low and $o_{auction}$ is early then p_{close} is low.
R_2:	If p_{ref} is low and $o_{auction}$ is late then p_{close} is very_low.
R_3:	If p_{ref} is medium and $o_{auction}$ is early then p_{close} is high.
R_4:	If p_{ref} is medium and $n_{auction}$ is small and $o_{auction}$ is late then p_{close} is medium.
R_5:	If p_{ref} is medium and $n_{auction}$ is big and $o_{auction}$ is late then p_{close} is low.
R_6:	If p_{ref} is high and $n_{auction}$ is small and $o_{auction}$ is early then p_{close} is very_high.
R_7:	If p_{ref} is high and $n_{auction}$ is big and $o_{auction}$ is early then p_{close} is high.
R_8:	If p_{ref} is high and $o_{auction}$ is late then p_{close} is medium.
R_9:	If p_{ref} is very_high and $o_{auction}$ is early then p_{close} is very_high.
R_{10}:	If p_{ref} is very_high and $o_{auction}$ is late then p_{close} is high.

- **Layer 1**: each node in this layer generates the membership degrees of a linguistic label for each input variable (*e.g.*, reference price is low or there are a big number of auctions left). Specifically, the ith node performs the following (fuzzification) operation:

$$O_i^{(1)} = \mu_{A_i}(x) = e^{-\frac{(x-c_i)^2}{2\delta_i^2}} + \gamma, \tag{1}$$

  where $O^{(1)}$ is the output of layer 1, x is the input to the ith node, and A_i is the linguistic value (small, medium, big, etc.) associated with this node. The set of parameters (c_i, δ_i) determines the shape of the membership function. These parameters can be adapted by learning. γ is a very small number to avoid the output in layer 1 being 0.
- **Layer 2**: each node calculates the firing strength of each rule (table I) via the multiplication operation:

$$O_i^{(2)} = w_i = \Pi_{j \in S_i^{(1)}} \{\mu_{A_j}\}, \tag{2}$$

  where $S_i^{(1)}$ is the set of nodes in layer 1 which feed into node i in layer 2, and w_i is the output of this node (*i.e.*, the strength of the corresponding rule).
- **Layer 3**: the ith node calculates the ratio of the ith rule's firing strength to the sum of all rules' firing strengths:

$$O_i^{(3)} = w_i' = \frac{w_i}{\sum_{j \in S^{(2)}} w_j}, \tag{3}$$

  where $S^{(2)}$ is the set of nodes in layer 2. This ratio indicates the relative importance of each rule.
- **Layer 4**: the ith output of the node is calculated by:

$$O_i^{(4)} = r_i \sum_{j \in S_i^{(3)}} w_j', \tag{4}$$

  where $j \in S_i^{(3)}$ is the set of nodes in layer 3 that feed into node i. The output of this layer combines all the outputs of the rules that have the same consequent output.
- **Layer 5**: the single node in this layer aggregates the overall output of the FNN (*i.e.*, p_{close}) as the summation of all incoming signals:

$$O^{(5)} = \sum_{i \in S^{(4)}} \left(r_i \sum_{j \in S_i^{(3)}} w_j' \right), \tag{5}$$

  where $j \in S^{(4)}$ is the set of nodes in layer 4.

C. FNN Learning

As discussed in section I, the FNN involves two types of learning: off-line and on-line. In either case, however, the same basic method is used. Specifically, the learning rule of the FNN agent is based on gradient descent optimisation [8]. Given the training data x_i ($i = 1, 2, 3$), the desired output value Y, and the fuzzy logic rules, the parameters of the membership functions for the FNN's input variables are adjusted by supervised learning. Here the goal is to minimise the error (E) function for all the training patterns:

$$E = \sum_j \frac{1}{2}(Y_j - O_j^{(5)})^2, \tag{6}$$

where Y_j is the actual closing price of pattern j and $O_j^{(5)}$ is the predicted closing price of the FNN for pattern j. For each set of training data, starting at the input nodes, a forward pass is used to compute the activity levels of all the nodes in the network. Then starting at the output nodes, a backward pass is used to compute $\frac{\partial E}{\partial O}$ for all the hidden nodes. Assuming that α is the adjustable parameter in a node, the general learning rule for adapting the FNN used is [6]:

$$\alpha(t+1) = \alpha(t) + \eta \left(-\frac{\partial E}{\partial \alpha} \right), \tag{7}$$

where η is the learning rate ($\eta \in [0.001, 0.01]$).

The FNN agent's parameters that get adjusted during learning are the consequent output of each rule (r_i in layer 4) and the centre and width of the Gaussian membership functions for each of the fuzzy terms (c_i and δ_i in layer 1). All of these parameters are adapted based on (7). Specifically, the adaptive rule of c_i in layer 1 is as follows (where $S_{-i}^{(2)}$ means the set of nodes in layer 2 that are connected with node i in layer 1):

$$\begin{aligned} \frac{\partial E}{\partial c_i} &= \sum_{m \in S_{-i}^{(2)}} \left(\frac{\partial E}{\partial O_m^{(2)}} \frac{\partial O_m^{(2)}}{\partial O_i^{(1)}} \frac{\partial O_i^{(1)}}{\partial c_i} \right) \\ &= \sum_{m \in S_{-i}^{(2)}} \left(\sum_{k \in S_{-m}^{(3)}} \left(\frac{\partial E}{\partial O_k^{(3)}} \frac{\partial O_k^{(3)}}{\partial O_m^{(2)}} \right) \frac{\partial O_m^{(2)}}{\partial O_i^{(1)}} \frac{\partial O_i^{(1)}}{\partial c_i} \right), \end{aligned} \tag{8}$$

where

$$\frac{\partial E}{\partial O_k^{(3)}} = (O_1^{(5)} - Y) r_k; \tag{9}$$

$$\frac{\partial O_k^{(3)}}{\partial O_m^{(2)}} = \begin{cases} \frac{\sum_k w_k - w_m}{(\sum_k w_k)^2} & \text{if k=m,} \\ \frac{-w_m}{(\sum_k w_k)^2} & \text{otherwise;} \end{cases} \tag{10}$$

$$\frac{\partial O_m^{(2)}}{\partial O_i^{(1)}} = \frac{w_m}{O_i^{(1)}}; \tag{11}$$

$$\frac{\partial O_i^{(1)}}{\partial c_i} = e^{-\frac{(x_i - c_i)^2}{2\delta_i^2}} \frac{(x_i - c_i)}{\delta_i^2}. \tag{12}$$

So the adaptive rule of c_i is:

$$c_i(t+1) = c_i(t) - \eta \frac{\partial E}{\partial c_i}. \tag{13}$$

Similarly, from (9), (10), and (11) the adaptive rule of δ_i is derived as:

$$\frac{\partial E}{\partial \delta_i} = \sum_{m \in S_{-i}^{(2)}} \left(\sum_{k \in S_{-m}^{(2)}} \left(\frac{\partial E}{\partial O_k^{(3)}} \frac{\partial O_k^{(3)}}{\partial O_m^{(2)}} \right) \frac{\partial O_m^{(2)}}{\partial O_i^{(1)}} \frac{\partial O_i^{(1)}}{\partial \delta_i} \right), \tag{14}$$

where

$$\frac{\partial O_i^{(1)}}{\partial \delta_i} = e^{-\frac{(x_i-c_i)^2}{2\delta_i^2}} \frac{(x_i-c_i)^2}{\delta_i^3}. \quad (15)$$

Hence the adaptive rule of δ_i becomes:

$$\delta_i(t+1) = \delta_i(t) - \eta \frac{\partial E}{\partial \delta_i}. \quad (16)$$

The learning rule for adjusting r_i in layer 4 and $(c_i,\ \delta_i)$ in layer 1 is:

$$\frac{\partial E}{\partial r_i} = \frac{\partial E}{\partial O^{(5)}} \frac{\partial O^{(5)}}{\partial O_i^{(4)}} \frac{\partial O_i^{(4)}}{\partial r_i} = (O^{(5)} - Y) \sum_{j \in S_i^{(3)}} w_j', \quad (17)$$

Hence r_i is updated by:

$$r_i(t+1) = r_i(t) - \eta (O^{(5)} - Y) \sum_{j \in S_i^{(3)}} w_j'. \quad (18)$$

D. Evaluating the Auctions

Given the expected auction closing prices, the agent needs to make a decision about which auctions to bid in. For ease of expression, we present this evaluation function for the case where only price and one other attribute of the good are considered. Given the user's preference on price and other attributes, the evaluations of the various factors need to be integrated. In fuzzy theory, the process of combining such individual ratings for an alternative into an overall rating is referred to as *aggregation* [9]. Let w_p and w_q be, respectively, the weight of price and the other attribute that the agent is concerned with, and u_p be the evaluation with respect to price and u_q the evaluation with respect to the other attribute. Intuitively, the role of the aggregation operator is to balance u_p and u_q and obtain an overall evaluation $u_{p,q}$ somewhere between the two values. There are three main aggregation operators that are commonly used and each of them has different semantics (conforming to different user objectives):

- Weighted average operator:

$$u_{p,q} = u_p w_p + u_q w_q, \quad (19)$$

  using this operator, even if one of the evaluations is very low, the overall output can still be reasonably high.
- Weighted Einstein operator [7]:

$$u_{p,q} = \frac{\frac{w_p}{\max\{w_p,w_q\}} u_p \times \frac{w_q}{\max\{w_p,w_q\}} u_q}{1 + \left(1 - \frac{w_p}{\max\{w_p,w_q\}} u_p\right)\left(1 - \frac{w_q}{\max\{w_p,w_q\}} u_q\right)} \quad (20)$$

  satisfies the characteristics of T-norms operators. Thus, if one evaluation is not satisfied (*i.e.*, $u_p = 0$ or u_q=0), the overall evaluation is 0. Intuitively, this corresponds to the situation where both evaluations must be satisfied.
- Uninorm operator [10]:

$$u_{p,q} = \frac{(1-\tau) u_p u_q}{(1-\tau) u_p u_q + \tau (1-u_p)(1-u_q)}, \quad (21)$$

  where $\tau \in (0,1)$ is the threshold of this operator. Thus, if both the evaluations are above the threshold, the overall evaluation is enhanced; if both are less than the threshold, the overall evaluation is weakened by each other; otherwise, the evaluation is a compromise if there is a conflict between the two evaluations.

Since these operators are all plausible means of aggregating price and the other attributes, and no one is necessarily superior in all cases, we need to empirically evaluate the impact of these operators on the performance of the agents. This we do in section IV-B.2.

III. Flight Auction Scenario

The scenario in this section provides an illustration of the operation of the FNN strategy and is the experimental domain for the evaluations reported in section IV. Here we consider how to model the user's preferences as fuzzy sets, outline the environmental setting for realising the scenario, and present the training results for the FNN agent.

In more detail, there are a number of airlines selling flight tickets through auctions. Each auction is selling a flight on a particular day. Each customer has an agent acting on their behalf and they are informed of the customer's preferences about price and travel date. The aim of the agent is to obtain the good that maximises the user's satisfaction degree.

A. Users' Preference Settings

We describe the valuation v of the agent for the flight ticket as a trapezoid shape fuzzy number. In this case, the higher the price of the good, the lower the satisfaction degree. When the price increases to the valuation of the agent, the satisfaction degree is 0. The travel date q is represented as a triangular fuzzy number. By way of illustration, suppose a customer's valuation about the ticket is about 300 pounds and she wants to travel on about the 15th of December. These preferences are expressed as fuzzy sets by the respective membership functions μ_P and μ_Q given as follows.

$$\mu_P(x) = \begin{cases} 1 & \text{if } x \le 200, \\ \frac{300-x}{100} & \text{if } 200 < x < 300, \\ 0 & \text{if } x \ge 300. \end{cases} \quad (22)$$

$$\mu_Q(y) = \begin{cases} \frac{y-12}{3} & \text{if } 12 \le y \le 15, \\ \frac{18-y}{3} & \text{if } 15 \le y \le 18, \\ 0 & \text{if } y \le 12 \text{ or } y \ge 18. \end{cases} \quad (23)$$

B. Experimental Settings

The experiments aim to cover a broad range of scenarios. All the parameters about the environment are assigned at the beginning of the game. Here we suppose that all auctions start at a price of 0 and all have a bid increment of 10 pounds. A day in the game equals 30 seconds of real time. An auction i's starting time and end time are randomly chosen from uniformly distributed ranges. Auction i's flight date and a customer's preferred travel date q_i are chosen randomly from $(11, 19)$ and $(12, 18)$ respectively. The valuation of the goods are randomly chosen from range $(170, 370)$.

C. The Learning Algorithm

As discussed in section II-C, the agent engages in a period of off-line learning in order to provide initial parameters for the FNN. Figure 2 shows the curve of the root mean square error with respect to the number of epochs. After 200 training epochs, it can be seen that the error between the target output and the actual output reaches its lowest point and so the

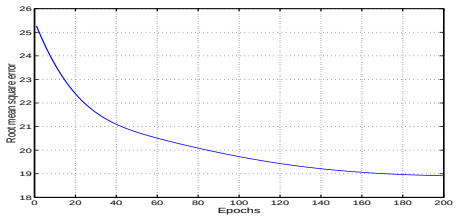

Fig. 2. Learning curve: root mean square error versus time.

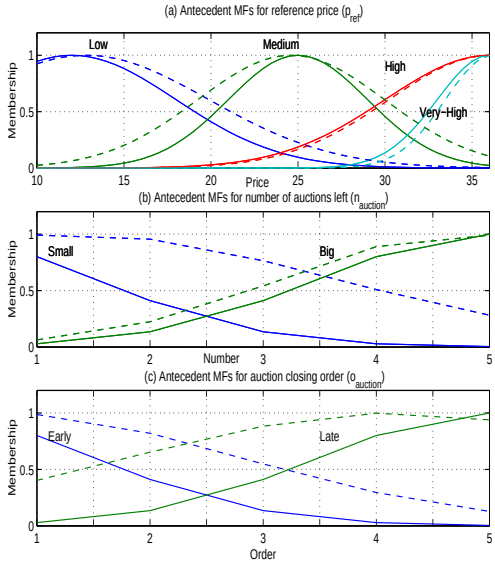

Fig. 3. Comparing antecedent membership functions (MFs) before (solid line) and after (dashed line) off-line learning.

parameters settings of this point are those used when the agent is made operational. Specifically, figures 3 shows the comparison of the FNN parameters before and after training. As can be seen, the parameters for each of the three inputs are adjusted from the original settings.

IV. Empirical Evaluation

This section evaluates the FNN agent by comparing it in a variety of environments, with other agents that use bidding strategies proposed in the literature.

A. Benchmarking Strategies

Since most of the extant multi-auction bidding strategies are concerned solely with price, we had to extend them to deal with bidding for goods that are characterised by multiple attributes. Thus, in all cases, the agents used the aggregation operators specified in section II-D in order to make trade-offs between price and travel date. The specific benchmark strategies we used are:

- *Greedy* strategy (adapted from [1]): unless the agent is active in some auction, bid in whichever auction currently has the highest evaluation (as defined in section II-D);
- *Fix Threshold* strategy (adapted from [2]): randomly assign an evaluation threshold $\theta \in [0, 1]$ for the satisfaction degree. Then bid in a randomly selected auction until the good is procured or θ is reached. In the latter case, switch to another random auction until all of them close.

TABLE II

Experimental results for varying agent populations. Each table cell gives a measure of how the agent performs in that setting and the percentage of that agent is in the population.

session no.	Agent Strategies			
	Greedy	Fix Auction	Fix Threshold	FNN
1	0.236 (40%)	0.186 (20%)	0.200 (20%)	0.297 (20%)
2	0.223 (20%)	0.174 (20%)	0.206 (40%)	0.313 (20%)
3	0.249 (20%)	0.173 (20%)	0.214 (20%)	0.286 (40%)
4	0.256 (25%)	0.165 (25%)	0.202 (25%)	0.312 (25%)
5	0.283 (20%)	0.186 (40%)	0.279 (20%)	0.361 (20%)

- *Fix Auction* strategy (adapted from [2]): Randomly select at the beginning of game the auctions in which bids will be placed and then only bid in these auctions. The auctions chosen here are those that have the highest satisfaction degree for date. The agent continues bidding in this auction until the price valuation is met, then it will switch to another auction.

B. Results

Three groups of experiments are conducted to evaluate the performance of each type of agent (sections IV-B.1 to IV-B.3). In each group of experiments, there are a number of sessions which correspond to experiments with different settings (as per section III-B). For each session, at least 200 games[1] are played among the agents.

Since the number of agents in each experiment varies, the performance ρ_K of a particular type K of agent (*i.e.*, Greedy, Fix Auction, Fix Threshold, and FNN) is calculated as:

$$\rho_K = \frac{\sum_i^{n_K} u_{p,q}^{(i)}}{n_K}, \tag{24}$$

where n_K is the number of type K agents in the same game and $u_{p,q}^{(i)}$ is the satisfaction degree for agent i.

1) **Varying Agent Populations:** This experiment aims to compare the performance of the different types of agents when there are varying numbers of the other agent types in the population (here the population size is fixed to 20). In this experiment, the weighted average operator is used and the weight ratio is $w_p : w_q = 1 : 1$. Table II shows the results when there are equal numbers of each type of agent, and when one type dominates numerically. From this, it can be seen that the FNN agents perform the best in all the situations considered. This success is due to their ability to be able to select and bid in a reasonable number of auctions that have a value close to the auction which has the maximum satisfaction degree. The Greedy agent is the next most effective. This agent endeavours to make a transaction whenever it can. Its main shortcoming is that it only considers ongoing auctions (it ignores those that have not yet started and so fails to consider the full set of potential purchasing opportunities when making bidding decisions). Thus, it sometimes buys a good at the user's valuation price, when, if it waited, it may well find subsequent auctions with lower closing prices. The Fix Auction performs worst because it only bids in auctions where it knows, a prior,

[1] A t-test showed that 200 games are sufficient to give a significant ranking among the agents. A *p value* of $p < 0.05$ is reported for all the experiments.

that it can get a high satisfaction on the flight date. This leads to a poor overall performance because it misses auctions that have a high evaluation on price but a lower one on date.

It can also be seen from table II that all the agents, except Fix Auction, behave best when there are many Fix Auction agents in the population. This is because the Fix Auction agents only bid in a small number of auctions. Thus other auctions have less competition and, consequently, lower prices. The FNN agent behaves worst when there are many agents like itself. This is because they tend to have similar predictions about the expected closing prices of the auctions and agents with the same preference date will compete strongly with one another in the same set of auctions.

2) **Varying Aggregation Operators:** This experiment studies the impact on the different types of agents of the different ways of trading-off the price and travel date (figure 4). To do this, we fix the number of auctions to be 10 and the number of agents to be 25. We also fix the weight of $w_p : w_q = 1 : 1$ for each operator. This time, the numbers of each agent type in a given game are randomly generated. Again, the FNN agents behave the best in all cases. In fact,

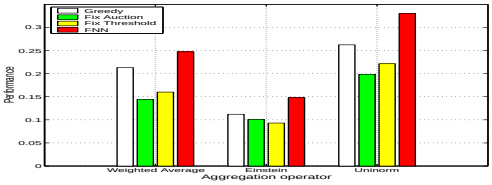

Fig. 4. Performance using different aggregation operators.

the order of performance of the four kinds of agent does not change from that reported above (for weighted average and uninorm operators). But for the weighted Einstein operator, the Fix Threshold agent is now the worst and the Fix Auction agent performs better. This is because for the Fix Auction agent, the evaluation for date is always high. Thus the overall evaluation of the auction is high; but for the other agents, the evaluation for date can be small.

3) **Varying Preference Weights:** This experiment evaluates the performance of the different agents when they use varying weights for the different attributes. This is an important issue to consider because, again, various weightings may lead to a different ranking of the strategies. Since the uninorm operator is usually used when $w_p = w_q = 0.5$. We do not consider it here. For weighted average and Einstein operators, we tested the impact of the weights on the performance of various strategies. In the experiment, we fix the number of auctions to be 10 and the number of agents to be 25 and figures 5 and 6 show the performance of the agents with the three representative weights we consider. In figure 5, the order of each kind of agent does not change for different weight ratios. However, the satisfaction degree decreases as w_q decreases. The reason for this is that the auctions' closing prices are always in the same broad range. Thus with the decrease of w_q, the evaluation on w_q also decreases, as does the overall satisfaction. Again, in all cases, FNN agents perform the best.

However, the order of the agents does change for different weight ratios when using an Einstein operator. As shown in

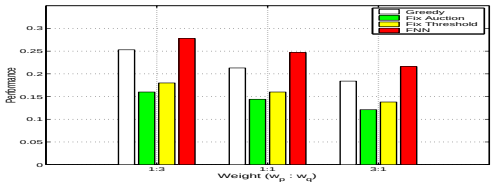

Fig. 5. Using weighted average operator with varying weights.

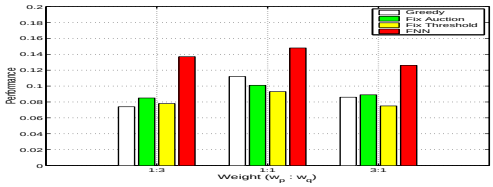

Fig. 6. Using weighted Einstein operator with varying weights.

figure 6, the Greedy agent's performance is worse than in the weighted average case. This is because the increasing evaluation for date is ignored since the agent only bids in those ongoing auctions in which it can make a profit. In contrast, the Fix Auction agent's performance is better than in the weighted average case due to its high evaluation of the date attribute. Again, in all cases FNN agents are the most successful.

V. Conclusions

This paper developed a new algorithm that guides an agent's bidding behaviour in multiple overlapping English auctions for a single item characterised by multiple attributes. The FNN strategy uses neuro-fuzzy techniques to predict the expected closing prices of the English auctions and to determine which auction the agent should bid in at what time. The use of a fuzzy neural network also allows the agent's decision making criteria to be adapted to the situation in which it finds itself. Moreover, we benchmarked our algorithm against a number of common alternatives available in the literature and showed the superior performance of our method. Our algorithm can also make trade-offs between the different attributes that characterise the desired good in order to maximise the user's satisfaction.

References

[1] A. Byde. A comparison among bidding algorithms for multiple auctions. Technical report, HP Research Labs, 2001.

[2] A. Byde, C. Preist, and N.R. Jennings. Decision procedures for multiple auctions. In *Proceedings of the first International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 613–620, 2002.

[3] M. He and N. R. Jennings. Designing a successful trading agent: A fuzzy set approach. *IEEE Transactions on Fuzzy Systems*, 2004.

[4] M. He, N. R. Jennings, and H. F. Leung. On agent-mediated electronic commerce. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):985–1003, 2003.

[5] M. He, H. F. Leung, et al. A fuzzy logic based bidding strategy for autonomous agents in continuous double auctions. *IEEE Transactions on Knowledge and Data Engineering*, 15(6):1345–1363, 2003.

[6] J. Jang. ANFIS: adaptive-network-based fuzzy inference systems. *IEEE Transactions on System, Man and Cybernetics*, 23(3):665–685, 1993.

[7] X. Luo, C. Zhang, and J. Cai. The weighting issue in fuzzy logic. *Informatica: An International Journal of Computing and Informatics*, 21(2):255–262, 1997.

[8] D.E. Rumelhart, G.E. Hinton, et al. Learning internal represenations by error propagation. *Parallel distributed processing*, 1:318–362, 1986.

[9] R.R. Yager. Aggregation operators and fuzzy systems modeling. *Fuzzy Sets and Systems*, 67:129–145, 1994.

[10] R.R. Yager and A. Rybalov. Uninorm aggregation operators. *Fuzzy Sets and Systems*, 80:111–120, 1996.