A Heuristic Bidding Strategy for Buying Multiple Goods in Multiple English Auctions

MINGHUA HE, NICHOLAS R. JENNINGS, and ADAM PRÜGEL-BENNETT
University of Southampton

This paper presents the design, implementation, and evaluation of a novel bidding algorithm that a software agent can use to obtain multiple goods from multiple overlapping English auctions. Specifically, an *Earliest Closest First* heuristic algorithm is proposed that uses neurofuzzy 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. This algorithm first identifies the set of auctions that are most likely to give the agent the best return and then, according to its attitude to risk, it bids in some other auctions that have approximately similar expected returns, but which finish earlier than those in the best return set. We show through empirical evaluation against a number of methods proposed in the multiple auction literature that our bidding strategy performs effectively and robustly in a wide range of scenarios.

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

General Terms: Algorithms, Design, Experimentation

Additional Key Words and Phrases: Intelligent agents, online auctions, multiple English auctions, bidding strategy, e-commerce

1. INTRODUCTION

Online auctions are increasingly being used for a variety of e-commerce applications [He et al. 2003]. This widespread adoption means there are invariably multiple auctions selling the desired good or service at the same time (for example, on eBay alone, there are typically over 13, 000 auctions for digital cameras at any one time). Moreover, it is frequently the case that there is a need to buy multiple goods from these multiple auctions. For example, buying a flight from one auction and booking a corresponding hotel from another or buying a car from a variety of cars for sale online. Given the fact that there is a huge search space, it is important to provide automated tools that can monitor relevant auctions, compare and make trade-offs between the offerings, decide which

M. He current affiliation is University of Wales, Bangor.
Authors' address: School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK; email: nrj@ecs.soton.ac.uk.

auctions to bid in, and determine what bids to place in the chosen auctions in order to obtain the best deals [Kephart 2002]. Software which can autonomously and flexibly achieve these tasks on behalf of a particular user is here termed a *software agent* [Jennings 2001].

Against this background, this article develops a heuristic bidding algorithm that a software agent can use to bid across multiple English auctions (with varying start and end times) in order to obtain multiple independent goods.[1] In this context, an English auction is one in which a single good is on offer and the auctioneer starts with his reserve (minimum acceptable) price and solicits successively higher (public) bids from the bidders and the last bidder remaining in the auction is the winner. We focus on English auctions because, although there are millions of different types of auction [Wurman et al. 2001] that can be used for e-commerce, the English auction is by far the most common [Lucking-Reiley 2000]. In our case, each such English auction is assumed to sell a single unit of the desired good and this good may be described by multiple attributes[2] (for example, in auctions selling flights, the goods can be described by their dates of departure and return, by their carrier, and the class of ticket being bought). We only consider the case where the agent buys multiple independent goods from such auctions; thus the failure to obtain one instance of the good does not influence the availability or desirability of other goods. Moreover, bids for these goods must be at least $h(a)$ pounds sterling (hereafter shortened to pound) larger than the previous price to be valid. Then if an agent bids successfully, it becomes the active agent that is holding the bid. It may, of course, be subsequently outbid. Auctions respond to any bid before they close and their good is allocated to the active bid holder when the auction closes.

In a stand-alone English auction where there is not a hard deadline, the bidding strategy is simple. The agent's dominant strategy (the best thing to do, irrespective of what the others do) is to bid the smallest allowable minimum amount more than the current highest bid and stop when the user's valuation is reached. By adhering to this dominant strategy, the good is always allocated to the bidder with the highest valuation. However, when there are multiple auctions running at the same time, bidding is much more complex and has much greater uncertainty. This is because (i) the auctions have varying start and end times (thus comparisons need to be made between auctions that are nearing completion (and probably have a high ask price) and auctions that are near the beginning (and probably have a low ask price); (ii) the participating agents are likely to adopt a variety of different strategies (for example, some may bid aggressively in the earlier auctions in order to ensure they get the desired goods, while others may wait until later auctions to see if they can obtain bargains), and (iii) the agents are likely to have different sets of

[1]The goods are actually substitutes in the sense that if the agent fails to buy an item in one of the auctions, it can bid in the other auctions if they are still running. Moreover, we assume the agent is tasked with obtaining multiple such goods from a number of different auctions. These multiple goods are independent in the sense that the failure to buy one of them does not influence the purchase of the others.

[2]The goods have multiple attributes, but the buyer agents bid only on price. Thus, our work differs from the case where agents bid on multiple attributes (as discussed in Section 6).

auctions that they consider bidding in (driven by their own deadlines), thus the set of agents in a given auction will vary and so, consequently, will the supply/demand.

In some variants of the English auction, there is a strict deadline for when the auction will finish and this promotes a strategy of trying to place a bid at the last moment so that no other agent has a chance to place a higher bid (this is called sniping [Rust et al. 1991]). To counteract such end effects, our auctions have a soft deadline; that is, they do not close until a fixed period after the last bid is placed (as per Yahoo!Auctions and Auction Universe). This means sniping is not effective and the auctions are akin to the standard English one. Moreover, we consider the case where the agent wants to purchase multiple goods from the ongoing auctions because this is a more general case than just purchasing a single item. In practice, an agent may require multiple goods either because a single owner wants multiple items or because multiple owners have combined forces in a form of group buying [Tsvetovat and Sycara 2000]. In the latter case, the agent also needs to allocate the goods it has purchased or is currently holding to the various customers in order to maximise its return.

Given this context, we have developed (for the first time) a heuristic bidding algorithm that buys multiple units of the desired good from the available auctions. This algorithm operates in the following way. It calculates what it believes are the best set of auctions to bid in (it does this by predicting the auctions' closing prices using a fuzzy neural network, allocating the goods to the customers it is acting on behalf of, and then calculating the satisfaction degree of the allocation). However the prediction of this best set of auctions is highly uncertain because it depends on the strategies, profiles, and reservation prices of an arbitrary set of agents. Therefore rather than just bidding in this set, an agent could also decide to bid in other auctions that are likely to have broadly the same outcome because, by doing so, its chances of obtaining the goods are increased (more places to buy from) and the likely satisfaction degree compared with what is believed to be the best ones is still reasonably high.

In more detail, our algorithm adopts the heuristic of bidding in the expanded set of auctions (the best set, plus those that have a broadly similar satisfaction degree) in the order of increasing auction end time. Thus we term it an *Earliest Closest First* (ECF) algorithm. Moreover, as the goods are composed of multiple attributes, the agent may have to make trade-offs between them in its bidding in order to best satisfy the user's preferences. Thus, for example, a user may ideally wish to fly out on a Saturday, return the following Wednesday and fly with British Airways, but would be willing to accept (for a lower price) flight dates of Friday and Wednesday with Quantas (a BA partner). To allow such flexibility (or imprecision), we choose to model preferences using fuzzy sets (since they have a proven track record for such tasks ([Luo et al. 2003, Zadesh 1965, 1975]). Specifically, the agent's preferences are private information that include (i) valuations v for the good (expressed as fuzzy sets); (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. By means of an example, consider the case of a student who wants to buy a flight ticket to New York. She prefers to buy a "cheap" ticket (cheap is a fuzzy term). Thus

the lower the price, the higher her degree of satisfaction. Ideally she wants to depart on "Saturday," but it is acceptable to go on Friday or Sunday. Here, the date can be denoted as a triangular fuzzy number [He et al. 2003], where Saturday has the highest satisfaction degree, and "Friday" and "Sunday" have lower ones. The airlines she likes can also be a fuzzy number where the memberships of the fuzzy set "like" are given by her satisfaction on the airlines. Finally, she can express the relative importance of the attributes of price, date, and airline by assigning them the appropriate weights.

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. Ideally the closing price of the auctions would be known first in order to calculate the satisfaction degree of a bid. However, since the closing price is only known after the auction is closed, it is important for the agent to make predictions about the likely closing prices of the various auctions. By so doing, the agent can determine whether it should place a bid at the current moment or it should delay because better deals might subsequently become available. In our previous work, we successfully used adaptive fuzzy inference methods for this task in continuous double auctions (CDA) [He et al. 2003] and the Trading Agent Competition (TAC) [He and Jennings 2004]. However, in both cases, the parameter adaptation of the fuzzy rules was somewhat limited. For example, our agent for the CDA can only adapt its parameters in a single direction of change (for example, all the parameters are bigger in a competitive environment). However this is inappropriate for the multi-auction context because each parameter in the strategy should be adjusted according to its actual direction of change (for example, the center of the membership function for the fuzzy set "medium" may need to go up, and the corresponding width may need to go down to reflect the fact that this fuzzy set should cover a smaller range of higher values). To rectify this, we exploit *fuzzy neural networks* (FNNs) [Jang 1993] since these can do the fuzzy reasoning and, through learning, can adjust the parameters of the fuzzy terms and the consequent output as the auctions progress. This adaptation enables the agent's bidding behavior to better reflect the current state of its environment (hereafter we call this strategy FNN and the agent using this strategy an FNN Agent[3]). We choose FNN for a number of reasons (1) As with a regular neural network, it can adapt its parameters to better fit the prevailing situation which cannot readily be determined a priori. (2) In contrast to a regular neural network, it is easier to put prior knowledge about the domain into the system (via the fuzzy rules), and this speeds up the learning process. (3) An FNN is easier to interpret than a standard neural network (because the output is obtained based on rules), and so it is easier to understand how and why the output is as it is.

The work described in this article advances the state of the art in the following ways. First, we develop for the first time an agent bidding algorithm (ECF)

[3]To clarify, an agent that uses the ECF bidding strategy needs to predict the auctions' closing prices. In our case, this is achieved via a FNN (although other methods could be used in the future). Indeed, in Section 5, we do use an alternative method for making this prediction, but retain the same ECF strategy as per the FNN agent benchmarking purposes.

for obtaining multiple goods from multiple overlapping English auctions. We believe this strategy is usable in a wide variety of e-commerce settings because it is based on readily observable information that can be found in many auction settings and because the underlying heuristic of bidding in auctions that have broadly similar expected returns is also widely applicable. Second, fuzzy neural networks are developed that can make predictions about the closing prices and adapt the parameters in the neural network through both offline and online learning to suit the environment the agent is situated in. Through empirical evaluation the agent that uses ECF combined with the FNN is shown to be effective in a wide range of situations. Finally, 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 the agent purchases and can cope with the inherent imprecision/flexibility that often characterises a user's preferences.

The rest of this article is structured as follows. Section 2 describes the ECF bidding algorithm, and Section 3 describes the FNN implementation. Section 4 gives an example of a flight auction scenario in which the operational effectiveness of our algorithm is evaluated. Section 5 actually provides the systematic empirical evaluation of the strategy and benchmarks it against a number of strategies that have been proposed in the literature. Section 6 discusses the related work in multiple auction bidding, multi-attribute auctions, and fuzzy-based bidding methods. Finally, Section 7 concludes and presents avenues for future research.

2. THE EARLIEST CLOSEST FIRST BIDDING ALGORITHM

This section details the ECF algorithm. First, however, we introduce some specific terms in our auction context and then the algorithm is described.

There are multiple auctions in the market, each selling one unit of good. There are three states for each auction (i) *waiting*: before its start time, nothing happens in this auction; (ii) *running*: the auction is open for bids; and (iii) *closed*: the auction finishes when the market time is bigger than the auction's closing time and there have been no active bids in the auction for a fixed period.

Each agent aims to buy multiple goods, thus it considers bidding if and only if the sum of the bids it holds (that is, those auctions in which it is the active bidder) and owns (closed auctions in which the agent won) is less than the number of good it desires.[4] If it decides to bid, the agent needs to determine which auctions it should bid in. To do so, it first determines the auctions that best satisfy the user's preferences (calculation is detailed in Section 3.1) given its expectation about the closing prices of each auction. Then, rather than placing a bid in the selected auctions immediately, it bids in auctions that close earlier than the selected auctions and have an evaluation "close" (a fuzzy term) to that of the selected auctions. The intuition here is that, given the significant degrees of uncertainty that exist, precise calculations about the closing prices are simply not reliable and an auction that appears slightly less promising may well

[4]The case where the agent bids in more auctions than the number of goods it wants is not considered here.

```
PROCEDURE ECFbid()
 1: while n_active + n_own < n_demand and AuctionRunning() do
 2:   update()                                  // get updated market information
 3:   predict()                                 // predict auctions' closing prices
 4:   allocate()                                          // allocate goods to user
 5:   for all g ∈ G do
 6:     if toBid(g) ≥ 0 then
 7:       s_best(g) ← evaluate(toBid(g), g)
 8:     else
 9:       s_best(g) ← 0
10:     end if
11:   end for
12:   L ← RunningAuctions()              // order the auctions in the closing time
13:   for all a ∈ L do
14:     for all g ∈ G do
15:       s ← evaluate(a, g)
16:       if toBid(g) ≥ 0 then
17:         λ ← chkThreshold()                          // decide the risk attitude
18:         if (s ≥ s_best(g)) or (s_best(g) − s ≤ λ) then
19:           bid(a)                                  // submit a bid in auction a
20:           break
21:         end if
22:       else if s > 0 then
23:         bid(a)
24:         break
25:       end if
26:     end for
27:   end for
28: end while
```

Fig. 1. The Earliest Closest First bidding algorithm.

turn out to be comparable or even 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, for each good it desires, if there are such close auctions, the agent will bid in the selected auctions in order of increasing closing time that is, bid first in the one that is going to close first, then in the one that will close next, and so on (hence the name Earliest Closest First). The degree of closeness that is required to trigger bidding is captured by the threshold ($\lambda \in [0, 1]$). Thus 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. If λ is high, the agent can be viewed as being risk averse because it bids in many more auctions in order to maximise its chance of getting the good (although it is likely to get a less satisfactory set of goods because it may accept a higher ask price). If λ is low, the agent is taking a greater risk because it is trying to obtain a high degree of satisfaction (but by not bidding in as many auctions, it has a lower chance of actually being successful). If λ is somewhere in between, the agent is striking a balance between the two positions.

In more detail, the decision making algorithm ECF is given in Figure 1. In this algorithm, n_{active}, n_{own}, and n_{demand} are the number of goods the agent holds, owns, and desires, respectively. An explanation of the algorithm's key functions is as follows.

—The function of *AuctionRunning* (line 1) returns *true* if there are still available auctions to bid in, *false* otherwise.

—The function of *update*() (line 2) 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.

—The function of *predict*() (line 3) predicts all auction's closing prices given the current market situation and the history transaction prices. The agent uses a FNN to predict the closing prices in this article (see Section 3.1).

—The function of *allocate*() (line 4) allocates all the goods the agent owns and possibly owns to its user according to their preferences. The later includes the goods the agent holds in waiting or running auctions. The way that we assign the auction to the user is through an assignment algorithm discussed in Section 3.4.

—The function of *toBid*(g) (lines 6, 7 and 16) returns the auction ID of the auction in which good g is believed to have the highest degree of satisfaction given the prediction of the closing prices. This is the output of the allocation in *allocate*(). If there is a best auction for buying good g given the allocation, the returned value will be the auction ID; otherwise, the agent will bid in any auction in which the current satisfaction degree is positive.

—The function of *RunningAuctions*() (line 12) returns a list L of all the currently running auctions in ascending order of their end times.

—The function of *evaluate*(a, g) (lines 7 and 15) returns the evaluation of auction a given the agents' preference for good g at its current ask price. This evaluation balances both price and the other attributes of the goods using a fuzzy aggregation method (see Section 3.3).

—The function of *chkThreshold*() (line 17) returns the threshold parameter for the agent given the current situation of the auction market. The threshold λ is determined by the number of auctions that have a positive satisfaction degree for the agent in the market at that particular moment in time. Here the general rule for choosing λ is: the more such auctions there are, the smaller λ should be. This captures the intuition that, if there are many chances for the agent to win the good, it can have a higher threshold so that it will have a higher satisfaction degree for the purchased goods (and vice versa). The experiments in Section 5.4 show the effect of different λs on the performance of the agent.

—The function of *bid*(a) (lines 19 and 23) places a bid in auction a. The price to place is the ask current price of the auction plus the bid step $h(a)$.

To realize this algorithm, a number of prediction techniques are needed (line 3). Here we use fuzzy neural networks (for the reasons outlined in Section 1) and their application is discussed in Sections 3.1 and 3.2. Moreover, an evaluation method is needed for ranking the various auctions (Section 3.3), and a good allocation method is needed to decide which auctions should be assigned to which user (Section 3.4). However, in other applications, a

FUNCTION $getRefPrice(i)$
1: **if** $getRelPrice(i) \geq 0$ **then**
2: Return $getRelPrice(i)$ *//return the average price in a single game*
3: **else**
4: Return $getAvgPrice(i)$ *//return the historical average price in past games*
5: **end if**

Fig. 2. Reference price (p_{ref}) calculation for the FNN. $getRel\,Price(i)$ returns the average transaction price of all the auctions (with the same attributes as auction i) that have closed since the agent started bidding. *getAvgPrice*(i) returns the historical average transaction price in the history for auctions (with the same attributes as auction i).

range of other technologies could be used to achieve the objectives of the ECF algorithm.

3. THE FNN IMPLEMENTATION

This section details the FNN implementation of the ECF algorithm. First, we describe how the FNN is structured and how it operates to obtain the predicted closing price. Second, the learning algorithm of the FNN is described. Third, the way of evaluating auctions is introduced. Finally, the allocation method that allocates the goods to the user is given.

3.1 FNN Prediction

To reason about the expected closing price of each auction, the FNN agent considers a per auction reference price (p_{ref}), the order in which the auctions are due to close ($o_{auction}$), and the number of alternative auctions ($n_{auction}$) where the similar item is available. Here the reference price represents a likely value at which the auction will close for that particular variant of the good [He et al. 2003]. It is computed by considering 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 (see function *getRefPrice*(i) in Figure 2). The FNN agent records such data examples and removes some of the oldest data to ensure it only learns based on the latest data. After each game finishes, the agent adjusts the parameters to better reflect the prevailing circumstances (see Section 3.2).

To predict the closing prices of the auctions, fuzzy reasoning is used. Through analyzing our experimental data, we found that the reference price, auction's closing order, and the number of substitute auctions are closely correlated with the actual closing prices.[5] Thus, fuzzy rules (defined in Table I) are designed to capture the relation among these factors. In particular, p_{ref} is expressed using the fuzzy linguistic terms high, medium and low; $o_{auction}$ is expressed by early, medium and late; and $n_{auction}$ is expressed by big, medium and small. The consequent output is expressed as the integer numbers very_high, high, medium and low.

Thus, the FNN agent takes three inputs (p_{ref}, $o_{auction}$, $n_{auction}$) and has one output (the expected auction closing price p_{close}). According to the rule base just

[5]In the future, we may consider adding other attributes, but our current analysis indicates these parameters are the main determinants of outcome.

Table I. The FNN Agent's Rule Base

R_1:	If p_{ref} is high and $o_{auction}$ is early and $n_{auction}$ is small then p_{close} is very_high.
R_2:	If p_{ref} is high and $o_{auction}$ is early and $n_{auction}$ is medium then p_{close} is very_high.
R_3:	If p_{ref} is high and $o_{auction}$ is early and $n_{auction}$ is big then p_{close} is high.
R_4:	If p_{ref} is high and $o_{auction}$ is medium and $n_{auction}$ is medium then p_{close} is high.
R_5:	If p_{ref} is high and $o_{auction}$ is medium and $n_{auction}$ is big then p_{close} is medium.
R_6:	If p_{ref} is high and $o_{auction}$ is late and $n_{auction}$ is medium then p_{close} is medium.
R_7:	If p_{ref} is high and $o_{auction}$ is late and $n_{auction}$ is big then p_{close} is low.
R_8:	If p_{ref} is medium and $o_{auction}$ is early and $n_{auction}$ is small then p_{close} is very_high.
R_9:	If p_{ref} is medium and $o_{auction}$ is early and $n_{auction}$ is medium then p_{close} is high.
R_{10}:	If p_{ref} is medium and $o_{auction}$ is early and $n_{auction}$ is big then p_{close} is medium.
R_{11}:	If p_{ref} is medium and $o_{auction}$ is medium and $n_{auction}$ is medium then p_{close} is medium.
R_{12}:	If p_{ref} is medium and $o_{auction}$ is medium and $n_{auction}$ is big then p_{close} is low.
R_{13}:	If p_{ref} is medium and $o_{auction}$ is late and $n_{auction}$ is medium then p_{close} is low.
R_{14}:	If p_{ref} is medium and $o_{auction}$ is late and $n_{auction}$ is big then p_{close} is very_low.
R_{15}:	If p_{ref} is low and $o_{auction}$ is early and $n_{auction}$ is small then p_{close} is high.
R_{16}:	If p_{ref} is low and $o_{auction}$ is early and $n_{auction}$ is medium then p_{close} is medium.
R_{17}:	If p_{ref} is low and $o_{auction}$ is early and $n_{auction}$ is big then p_{close} is low.
R_{18}:	If p_{ref} is low and $o_{auction}$ is medium and $n_{auction}$ is medium then p_{close} is low.
R_{19}:	If p_{ref} is low and $o_{auction}$ is medium and $n_{auction}$ is big then p_{close} is very_low.
R_{20}:	If p_{ref} is low and $o_{auction}$ is late and $n_{auction}$ is medium then p_{close} is very_low.
R_{21}:	If p_{ref} is low and $o_{auction}$ is late and $n_{auction}$ is big then p_{close} is very_low.

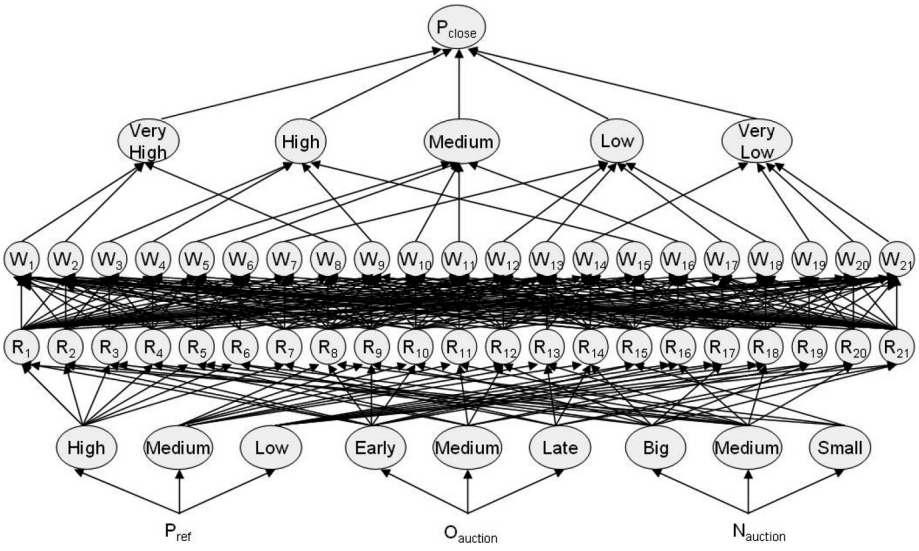

Fig. 3. Overview of the FNN architecture.

mentioned, we developed a FNN with 5 layers (as shown in Figure 3). The input variables correspond to the nodes in layer 1. The nodes in layer 2 correspond to individual rules for reasoning about the closing prices of the auctions. Nodes in layer 3 calculate the relative importance (weight) of each of these rules. The nodes in layer 4 combine the output of each rule into the overall output. Finally, the node in layer 5 sums up all the outputs in layer 4 and gives the predicted

auction closing price. In more detail:

—*Layer* 1. Each node in this layer generates the membership degrees of a linguistic label for each input variable (for example, reference price is low or there are a large number of substitute auctions in the market). 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 (that is, the membership degree with respect to the corresponding fuzzy sets), x is the input to the ith node, and A_i is the linguistic value (high, medium, low, etc.) associated with this node. The set of parameters (c_i, δ_i) determines the shape of the membership function.[6] These parameters can be adapted by learning (as we will explain in Section 3.2). γ is a very small number (here we choose 0.00001) that avoids the output in layer 1 from becoming zero.

—*Layer* 2. Each node in this layer calculates the firing strength (the mimimum of all the inputs, and it is in the range of [0, 1]) of each rule (in 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 (that is, the strength of the corresponding rule).

—*Layer* 3. The ith node of this layer 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 (that is, 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.

[6]Here we assume that this is a Gaussian function. This is because it has nonzero derivatives throughout the universe of discourse and is therefore easy to implement. Also, the derivatives of a Gaussian function are continuous and smooth, thus, it can produce good training performance.

Given the nature of this decision making task, it is important that the various parameters of the FNN algorithm fit the prevailing context as accurately as possible. This is achieved through combined learning (see Section 3.2 for more details). To start, a number of simulated games[7] 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. The agent keeps track of the various auctions and records the latest data which is then combined with the old data. The combined data set is fed into the FNN as new training examples. However, these examples are weighted more highly than older ones and so enable the agent to better reflect prevailing circumstances (but without being completely reactionary to the last set of changes).

To illustrate the operation of this architecture consider the following example. Let the reference price for the next auction to bid in be $p_{ref} = 20$, and assume that it will be the fourth one to close $o_{auction} = 4$. Suppose there are 8 substitute auctions. Thus $(20, 4, 8)$ is fed into the FNN. Let these values be assigned the following membership degrees of the following fuzzy sets: *medium* $(20) = 0.8$, *late* $(4) = 0.7$ and *big* $(8) = 0.5$ (from (1)). These values are then the respective outputs of nodes 2, 6 and 7 in layer 1. Then, taking R_{14} as an example, the output of node R_{14} in layer 2 is $0.8 \times 0.7 \times 0.5 = 0.28$ (from (2)). Then, suppose the sum of all the firing strengths in layer 2 is 3. The weight of R_{14} (output of w_{14}) will be $0.28/3 = 0.093$ (from (3)). Thus, R_{14} contributes 0.093 among all the rules. After this, in layer 4, there are 3 rules that have the same consequent output, which is *very_low*, that is, R_{14}, R_{19}, R_{20}, and R_{21} (from Figure 3). Let $r_{very\_low} = 12$, $w_{19} = 0.1$, $w_{20} = 0.1$, and $w_{21} = 0.1$. In which case, the output of node *very_low* is $12 \times (0.093 + 0.5 + 0.2 + 0.1) = 10.72$ (from (4)). Finally, the output of layer 5 is the sum of all the outputs of layer 4 (using Equation (5)) one of which will be the 10.72 coming from the very_low node.

Finally, to guarantee the predicted closing price is alway higher than the current ask price, the predicted auction closing price will be

$$P^*_{close} = \max\{p_{close}, p_{now}\}, \tag{6}$$

where p_{close} is the output of the FNN and p_{now} is the current ask price of the auction.

3.2 FNN Learning

Given the training data x_i $(i = 1, 2, 3)$, the desired output value Y, and the fuzzy logic rules (from Table I), the parameters of the membership functions for the FNN's input variables are adjusted by supervised learning. Here the

[7]The initial parameters of the FNN can either be set directly by the user or can be set by playing simulated games. A simulated game is played by strategies or humans in a test environment where the various agents compete but money does not actually change hands. In the cases where such games are not available, users can monitor real auction Web sites (such as eBay) and collect relevant data (since the trading history of most auction sites is readily available). However, if the user is confident about their parameter settings, the agent can be put directly into practice without going through the simulated games.

goal is to minimize the error (E) function for all the trainings

$$E = \sum_j \frac{1}{2}\left(Y_j - O_j^{(5)}\right)^2, \tag{7}$$

where Y_j is the actual closing price of pattern[8] 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. In our FNN agent, the parameters that get adjusted during learning are the consequent output of each rule (r_i in layer 4) and the center and width of the Gaussian membership functions for each of the fuzzy terms (c_i and δ_i in layer 1). For the parameters of r, the learning rule of the FNN agent is based on standard gradient descent optimization [Rumelhart et al. 1986]

$$r(t+1) = r(t) + \eta\left(-\frac{\partial E}{\partial r}\right), \tag{8}$$

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

Thus, the learning rule for adjusting the parameters of r_i in layer 4 and (c_i, δ_i) in layer 1

$$\frac{\partial E}{\partial r_i} = \frac{\partial E}{\partial O^{(5)}}\frac{\partial O^{(5)}}{\partial O_i^{(4)}}\frac{\partial O_i^{(4)}}{\partial r_i} = (O^{(5)} - Y)\sum_{j \in S_i^{(3)}} w_j', \tag{9}$$

Hence r_i is updated by:

$$r_i(t+1) = r_i(t) - \eta(O^{(5)} - Y)\sum_{j \in S_i^{(3)}} w_j'. \tag{10}$$

For parameters c_i and s_i, the conjugate gradient algorithm [Johansson et al. 1990] is used since it is shown to be faster than the standard gradient descent optimization [Rumelhart et al. 1986]. Here suppose α is the parameter we are interested in and the gradient of iteration t of the learning is g_t ($t > 1$), then the new search direction is to combine the new steepest descent direction with the previous one, that is,

$$p_t = -g_t + \beta_t p_{t-1}, \tag{11}$$

where by using Fletcher-Reeves [Fletcher and Reeves 1964] update,

$$\beta_t = \frac{g_t^T g_t}{g_{t-1}^T g_{t-1}}. \tag{12}$$

Thus, 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):

$$\frac{\partial E}{\partial c_i} = \sum_{m \in S_{-i}^{(2)}} \left(\frac{\partial E}{\partial O_m^{(2)}}\frac{\partial O_m^{(2)}}{\partial O_i^{(1)}}\frac{\partial O_i^{(1)}}{\partial c_i}\right)$$

[8]Here a pattern is a training example (for example, for the previous example, a pattern might be {{20, 4, 8}, 10.722}, where given the inputs {20, 4, 8} the actual output is a price of 10.722).

$$= \sum_{m \in S_{-i}^{(2)}} \left(\sum_{k \in S_{-m}^{(3)}} \left(\frac{\partial E}{\partial O_k^{(3)}} \frac{\partial O_k^{(3)}}{\partial O_m^{(2)}} \right) \frac{\partial O_m^{(2)}}{\partial O_i^{(1)}} \frac{\partial O_i^{(1)}}{\partial c_i} \right), \tag{13}$$

where

$$\frac{\partial E}{\partial O_k^{(3)}} = (O_1^{(5)} - Y) r_k; \tag{14}$$

$$\frac{\partial O_k^{(3)}}{\partial O_m^{(2)}} = \begin{cases} \frac{\sum_k w_k - w_m}{(\sum_k w_k)^2} & \text{if k = m,} \\ \frac{-w_m}{(\sum_k w_k)^2} & \text{otherwise;} \end{cases} \tag{15}$$

$$\frac{\partial O_m^{(2)}}{\partial O_i^{(1)}} = \frac{w_m}{O_i^{(1)}}; \tag{16}$$

$$\frac{\partial O_i^{(1)}}{\partial c_i} = e^{-\frac{(x_i - c_i)^2}{2\delta_i^2}} \frac{(x_i - c_i)}{\delta_i^2}. \tag{17}$$

So the adaptive rule of c_i is:

$$c_i(t+1) = c_i(t) + \eta p_t, \tag{18}$$

where $p_t = -g_t + \beta_t p_{t-1}$ and $g_t = \frac{\partial E}{\partial c_i}$.

Similarly, from Equations (14), (15), and (16), the adaptive rule of δ_i is derived as:

$$\frac{\partial E}{\partial \delta_i} = \sum_{m \in S_{-i}^{(2)}} \left(\sum_{k \in S_{-m}^{(3)}} \left(\frac{\partial E}{\partial O_k^{(3)}} \frac{\partial O_k^{(3)}}{\partial O_m^{(2)}} \right) \frac{\partial O_m^{(2)}}{\partial O_i^{(1)}} \frac{\partial O_i^{(1)}}{\partial \delta_i} \right), \tag{19}$$

where

$$\frac{\partial O_i^{(1)}}{\partial \delta_i} = e^{-\frac{(x_i - c_i)^2}{2\delta_i^2}} \frac{(x_i - c_i)^2}{\delta_i^3}. \tag{20}$$

Hence the adaptive rule of δ_i becomes

$$\delta_i(t+1) = \delta_i(t) - \eta p_t, \tag{21}$$

where $p_t = -g_t + \beta_t p_{t-1}$ and $g_t = \frac{\partial E}{\partial \delta_i}$.

3.3 Evaluating the Auctions

Given the expected auction closing prices, the agent needs to make a decision about which auctions to bid in.[9] For ease of expression, we present this evaluation function for the case where only price and one other attribute of the good are considered (but the concepts are equally applicable for arbitrary numbers of attributes). Given the user's preference on price and other attributes (as defined in Section 1), the evaluations of the various factors need to be integrated. In fuzzy theory, the process of combining such individual ratings for

[9]Such an evaluation function is used to evaluate the bidding strategy that considers more than one of the good's attributes in making its bidding choice. Thus, for example, all the benchmark strategies in Section 5 exploit such a function.

an alternative into an overall rating is referred to as *aggregation* [Yager 1994]. Now 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

$$u_{p,q} = u_p w_p + u_q w_q. \tag{22}$$

Using this operator means that even if one of the evaluations is very low, the overall output can still be reasonably high. For example, if the user does not like the time of the flight but it is very cheap, the overall evaluation can still be high.

—Weighted Einstein operator [Luo et al. 2003]

$$u_{p,q} = \frac{u_p^{'} u_q^{'}}{1 + (1 - u_p^{'})(1 - u_q^{'})}, \tag{23}$$

where $u_p^{'} = (u_p - 1)w_p$, and $u_q^{'} = (u_q - 1)w_q$. This equation ensures that if one evaluation is not satisfied (that is, $u_p = 0$ or $u_q = 0$), the overall evaluation is 0. Intuitively, this corresponds to the situation where both evaluations must be satisfied more or less. For example, even if the flight ticket is free, the user cannot accept it since traveling after a specific date is totally useless (for example, he has a very import meeting at a specific date).

—Weighted uninorm operator [Yager and Rybalov 1996]

$$u_{p,q} = \frac{(1 - \tau) u_p^{'} u_q^{'}}{(1 - \tau) u_p^{'} u_q^{'} + \tau (1 - u_p^{'})(1 - u_q^{'})}, \tag{24}$$

where $u_p^{'} = \frac{(u_p - 1)w_p}{\max\{w_p, w_q\}} + 1$, and $u_q^{'} = \frac{(u_q - 1)w_q}{\max\{w_p, w_q\}} + 1$, $\tau \in (0, 1)$ is the unit element of this operator. The unit element can be regarded as a threshold: 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; if there is a conflict between the two evaluations, the overall evaluation is a compromise. For example, if the user likes the date and price, the overall evaluation is very high; if the user hates the date and price, the overall evaluation is even lower; and if the user likes the date but hates the price, then some intermediate value is chosen.

Since these operators are all plausible means of aggregating price and the other attributes, and none 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 5.2.

3.4 Goods Allocation

The agent needs to allocate the goods it owns and potentially owns to its customers in order to maximize the overall satisfaction degree (if there is a single

customer, then this is a trivial stage). Thus good allocation takes place each time the agent accesses the market and when the market information has been updated. The goods are allocated to the agent's users optimally so as to maximize the sum of the users' satisfaction. Here this allocation process is regarded as an assignment problem which we solve using a shortest augmenting path algorithm [Jonker and Volgenant 1987] (this is commonly used to solve assignment problems because of its stability and efficiency).

In more detail, suppose D is a $d \times d$ square,[10]

$$D = \begin{pmatrix} s_{11} & \cdots & s_{1d} \\ & \cdots & \\ s_{d1} & \cdots & s_{dd} \end{pmatrix},$$

where d is the number of auctions, and s_{ij} is the satisfaction degree of the user's jth requirement for auction a_i:[11]

$$s_{ij} = \begin{cases} -evaluate(a_i, g_j) & \text{if } i \le n_{demand}, \\ 99 & \text{if owns or holds a good in } a_i \text{ and } j > n_{demand} \\ 0 & \text{otherwise,} \end{cases}$$

where 99 is used to avoid the owned goods being allocated to the dummy nodes.

Given this input, suppose an allocation X is

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1d} \\ & \cdots & \\ x_{d1} & \cdots & x_{dd} \end{pmatrix},$$

where each row and each column has only a single 1 (that is, each auction is only allocated to one user). Thus,

$$\sum_{j=1}^{d} x_{ij} = 1,$$

$$\sum_{i=1}^{d} x_{ij} = 1,$$

and the objective function to minimize is

$$\sum_{j=1}^{d} \sum_{i=1}^{d} (s_{ij} x_{ij}),$$

where $x_{ij} = 0$ or 1.

Using this method, the agent can decide how to allocate the goods it owns and holds to its users optimally given the ask price or the predicted price of the auctions. This assignment method is also used to calculate the performance of the agents at the end of the game.

[10]Here a square is necessary in order to use the algorithm. The row represents the auction, and the column represents the goods the agent desires. To use the algorithm, some dummy nodes may need to be added to make a square so that the row number is equal to the column number.

[11]Our problem here is a maximization problem, but in order to use the shortest augmenting path algorithm, we need to put a minus before the evaluation value.

4. FLIGHT AUCTION SCENARIO

This section provides an intuitive scenario[12] in which the operation of our ECF algorithm can be exemplified and its performance empirically assessed (see Section 5). Here we consider how to model the user's preferences as fuzzy sets, outline the environmental setting for realizing 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 one flight ticket. Each software agent is acting on behalf of one customer, and they are informed of the customer's preferences about prices and travel dates.[13] The aim of the agent is to obtain the goods that maximize the sum of its users' satisfaction.

4.1 Users' Preference Settings

We describe the valuation v of a customer for a flight ticket as a trapezoid shape fuzzy number $(l_{bottom}, l_{top}, r_{top}, r_{bottom})$, where $l_{bottom} = 0$ and $l_{top} = 0$, and r_{top} and r_{bottom} are the values where the satisfaction starts to decrease and where it becomes 0. 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[14] (l_q, c_q, r_q), where c_q is the preferred date and l_q and r_q are the left and right limits, respectively.[15]

By way of illustration, suppose a customer's valuation for the ticket is about 300 pounds and she wants to travel on or about the 15th of December. These preferences are expressed as fuzzy sets by the respective membership functions μ_P and μ_Q given in Equations (25) and (26) and are shown graphically in Figures 4 and 5.

$$\mu_P(x) = \begin{cases} 1 & \text{if } x \leq 100, \\ \frac{300-x}{200} & \text{if } 100 < x < 300, \\ 0 & \text{if } x \geq 300. \end{cases} \tag{25}$$

$$\mu_Q(y) = \begin{cases} \frac{y-12}{3} & \text{if } 12 \leq y \leq 15, \\ \frac{18-y}{3} & \text{if } 15 \leq y \leq 18, \\ 0 & \text{if } y \leq 12 \text{ or } y \geq 18. \end{cases} \tag{26}$$

[12]We choose (for reasons of familiarity) a flight auction scenario where an agent is trying to buy multiple flight tickets on behalf of a user. This is a problem that we often meet in real life and it fits the requirements of our context as outlined in Section 1. It has also been used in the Trading Agent Competition which is an international forum for benchmarking bidding strategies (see Section 6 for more details).

[13]For reasons of simplicity, we focus on the two attribute case. However, the principle is similar with more attributes.

[14]Any kind of fuzzy number can be used here, for example, trapezoid or bell-shaped fuzzy numbers. We choose a triangular one simply because it is the most commonly used.

[15]If a user has a crisp preference, for example, he has to travel on the 15th of December, the similarity degree of the 15th is 1 and 0 otherwise.

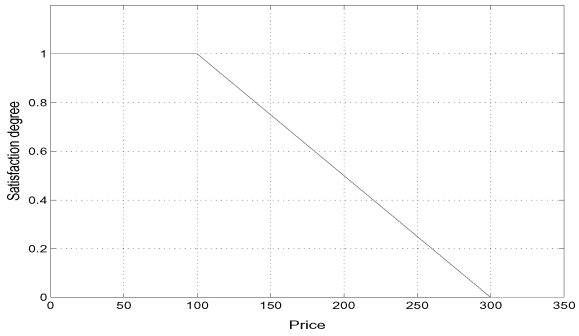

Fig. 4. Customer's preference about price.

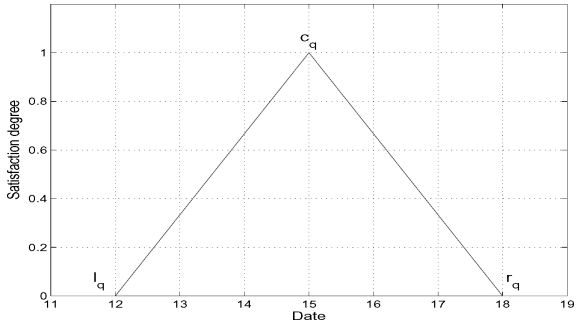

Fig. 5. Customer's preference about travel date.

4.2 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 100, and all have a bid increment of 10 pounds. Also

—a day in the game equals $\zeta = 5$ seconds of real time[16];

—an auction i's starting time t_i^{start} is randomly chosen from a uniformly distributed range $(0, (q_i - 5)\zeta)$. This ensures all the auctions start at least five days before the travel date[17];

—auction i's end time is randomly chosen from a uniformly distributed range $(t_i^{start} + 2\zeta, (q_i - 3)\zeta)$. This guarantees that the auctions close at least three days before the travel date[18];

—each agent is assigned to a customer which has n requirements, $n \in [1, 8]$;

[16]The length of one day can be shorter if it can be guaranteed that all the agents have time to respond in the market.

[17]We choose five to ensure the agent has a reasonable time to transact the ticket before the travel date.

[18]We choose three to ensure that the start time of the auction is before the end time, and there is a reasonably long time for the auction.

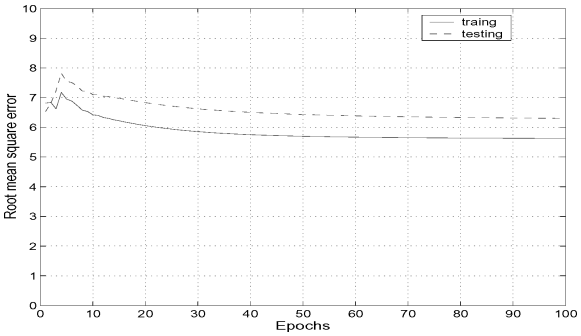

Fig. 6. Learning curve: root mean square error versus time.

—all the agents start bidding at the beginning of the game[19];

—auction i's flight date q_i is chosen randomly from a uniformly distributed range (11, 19);

—the valuation of the goods for a customer are randomly chosen from a uniformly distributed range (170, 370);

—a customer's preferred travel date is randomly chosen from a uniformly distributed range (12, 18)[20].

4.3 The FNN Agent's Learning Algorithm

As discussed in Section 3.2, the agent engages in a period of offline learning in order to provide initial parameters for the FNN agent. In more detail, Figure 6 shows the training and testing curves of the root mean square error with respect to the number of training epochs.[21] The errors are computed for both the training set and the test set. After each game is played, the new game data (test data) and the latest game data before this game (training set) are input to the FNN for learning. After 100 training epochs, it can be seen that the error between the target output and the actual output reaches its lowest point and so the parameters settings of this point are those used when the agent is made operational. Specifically, Figures 7 and 8 show, respectively, 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 defined by the field experts.

4.4 Parameter Adaptation in Different Environments

This section compares the parameter adaptation in two environments where the supply is high (25 auctions) and low (15 auctions). Specifically, Figures 9

[19]This is because we want to evaluate all types of agents fairly. If some agents start bidding late, they will be at a disadvantage compared with those who start bidding early.

[20]This range is smaller than the range of the auctions' flight dates because this preferred travel date is a fuzzy number. Thus when defuzzified, it will actually cover the full range of the flight's dates.

[21]The reference price and ask price are scaled by dividing by 10 during learning. However, this does not affect the result in any way. In the real-world scenario, the price can be any number.

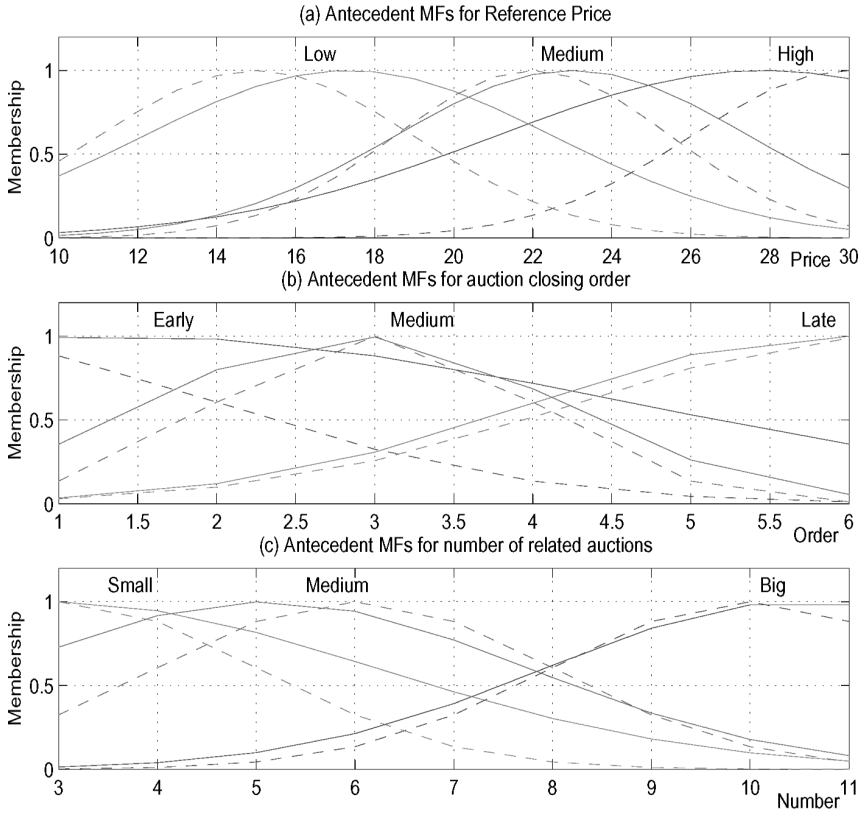

Fig. 7. Comparing antecedent membership functions (MFs) before (dashed line) and after (solid line) offline learning.

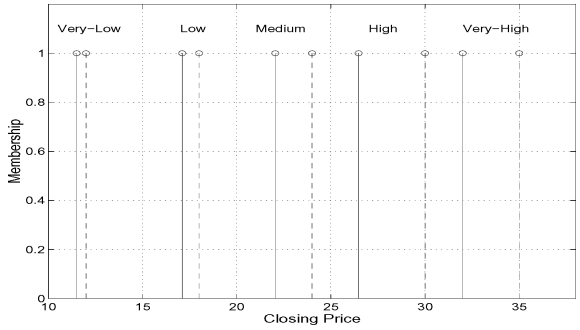

Fig. 8. Comparing consequent membership functions before (dashed line) and after (solid line) offline learning.

to 12 show how the parameters are adjusted differently in different environments. In Figure 10, when supply is high, the closing prices tend to be low and, thus, the consequent parameters are lower than the initial ones. In contrast, in Figure 12, when supply is low, the closing prices tend to be high, and the consequent parameters are higher than the initial ones.

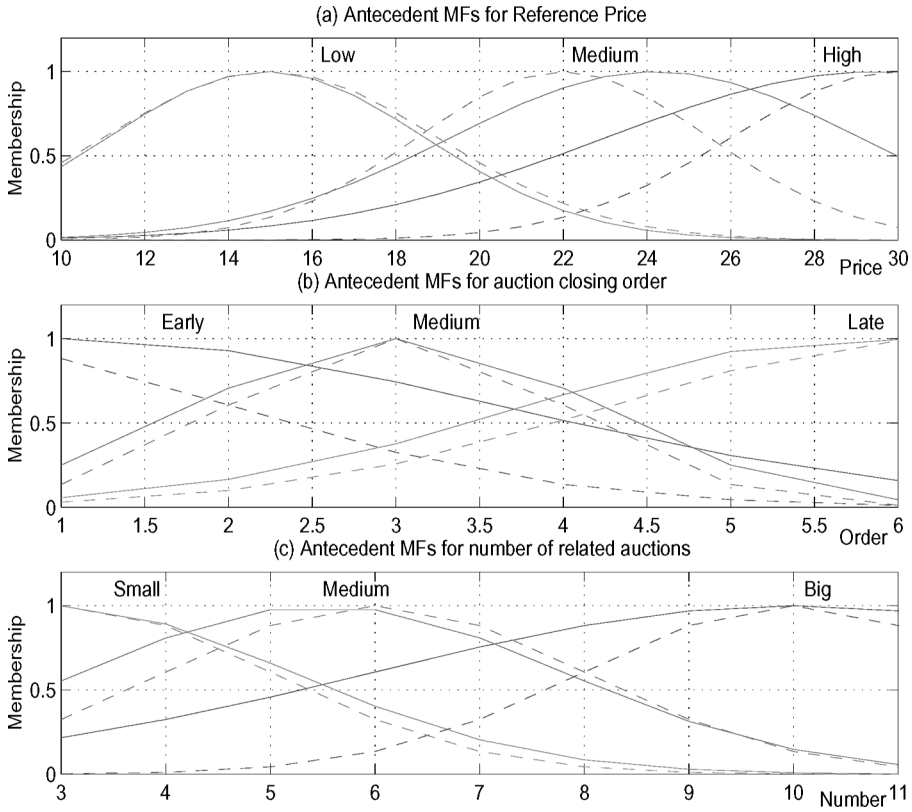

Fig. 9. Comparing antecedent membership functions (MFs) before (dashed line) and after (solid line) offline learning in-high supply environment.

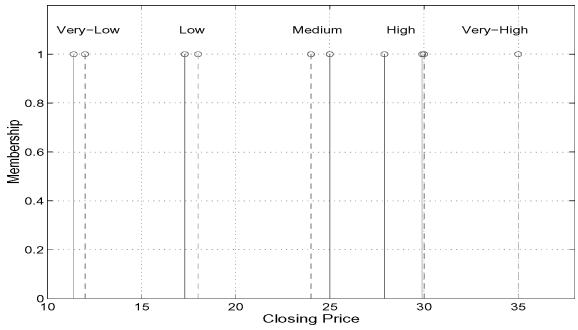

Fig. 10. Comparing consequent membership functions before (dashed line) and after (solid line) offline learning in a high supply environment.

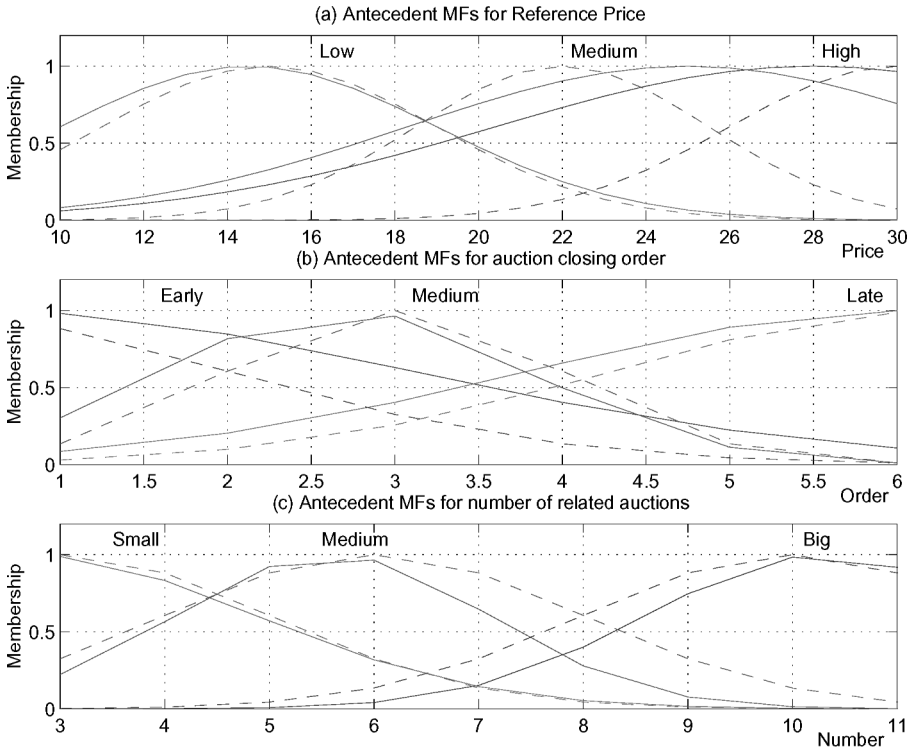

Fig. 11. Comparing antecedent membership functions (MFs) before (dashed line) and after (solid line) offline learning in a low supply environment.

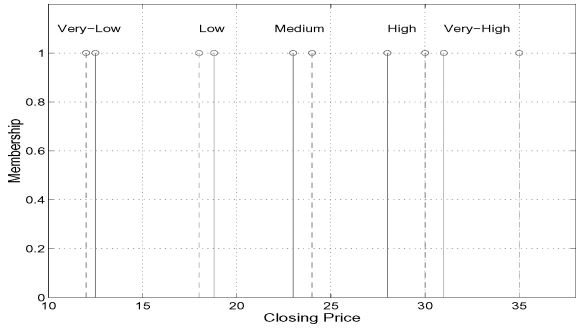

Fig. 12. Comparing consequent membership functions before (dashed line) and after (solid line) offline learning in a low supply environment.

5. 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. In particular, we are interested in assessing the performance of each kind of agent in different environments. There are three main groups of experiments, and there are a number of sessions which correspond to experiments with

different settings (as per Section 4.2). For each session, at least 200 games[22] 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 (for example, FNN) is calculated as the average satisfaction degree per agent of the kind K, that is:

$$\rho_K = \frac{\sum_i^{n_K} \sum_j^{m_i} u_j^{(i)}}{n_K}, \tag{27}$$

where n_K is the number of type K agents in the same game, $u_j^{(i)}$ means the evaluation of customer j, and m_i is the number of customers of agent i. Since most of the extant multi-auction bidding strategies are concerned solely with price (see Section 6), we had to extend them to deal with bidding for goods that are characterized by multiple attributes. Thus, in all cases, the agents used the aggregation operators specified in Section 3.3 in order to make trade-offs between price and travel date. To deal with multiple goods, the allocation function can decide which user bids in which auction. The specific benchmark strategies we used are the following.

—*Greedy* (GRD) *Strategy* (adapted from Byde [2001a]). While the sum of the number of goods held and owned is less than what the agent needs, bid in auctions where the auction's current satisfaction has the highest evaluation (as defined in Section 3.3);

—*Fixed Auction* (FIX) *Strategy* (adapted from Byde et al. [2002]). Select at the beginning of the game the auctions in which bids will be placed, and then only bid in these auctions. The auctions chosen here are those where the sum of the users' satisfaction is highest for date (at this time none of them have a value for price). The agent continues bidding in its selected auction until the price satisfaction degree equals zero, in which case it will switch to another auction (until all those in the fixed set have been tried).

—*Average* (AVG) *Strategy*. AVG also uses the ECF algorithm, but it uses a much simpler prediction function based on the past history transaction prices to predict the closing prices (that is, to implement the *predict()* function in Figure 1). In more detail, it calculates the average closing prices of all the auctions for each kind of good from the recent games. Suppose in the latest N games, there are m auctions with attribute i, then the predicted closing price of an auction with attribute i is

$$\tilde{p}_{close}^{(i)} = \frac{\sum_j^m p_j^{(i)}}{m},$$

where $p_j^{(i)}$ is the real closing price of auction j with attribute i.

5.1 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). In this experiment, we studied three

[22]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.

environments: when supply is low (15 auctions), medium (20 auctions), and high (25 auctions). When the weighted average operator is used and the weight ratio is $w_p : w_q = 1 : 1$, Figure 13 shows the results when there are fixed numbers of each type of agent in a session (a), and when one type dominates numerically (b) to (e).[23]

From this, it can be seen that the FNN agents perform better than other agents in all the cases considered. We attribute this success to their ability to be able to select the auctions to bid in according to the relatively correct prediction on the closing prices of the auctions. In more detail, the FNN agent is better than GRD agents. This is because the GRD agent endeavors 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 had waited, it might well find subsequent auctions with lower closing prices. The FIX agent performs the worst because it only bids in auctions where it knows a priori that it can get 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. As is shown in Figure 13, FIX agents have the smallest transaction numbers.

It can also be seen from all the subfigures in Figure 13 that, when the supply is high, all the agents have a higher performance value than when there is a low supply. This is as expected because, in general, the auctions close at a lower price when the supply is high (because there is less compitition). In Figure 13(b), GRD agents dominate the market, and they often make the transaction price of some auctions very high. Moreover, it can be seen from the performance in Figure 13(b), when compared with other figures in Figure 13, that GRD agents have a relatively worse performance value. When FIX agents dominate the market (Figure 13(d)), all the other agents have a higher performance value compared to the other cases in Figure 13. This is because the FIX agents only bid in a small number of auctions. Thus other auctions have less competition and, consequently, lower prices.

5.2 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[24] (see Figure 14). To do this, the number of auctions is generated randomly in the range of [15, 25] and the number of agents is 8. We also fix the weight of $w_p : w_q = 1 : 1$ for each

[23]In (a), there are equal numbers of each agent, and we have 4 kinds of agents. There are 4 agents and each agent desires 8 units of goods. In (b) to (e), there are 2 of one kind of agent and 1 of the other three kinds. Thus, there are 5 agents in total, and each agent desires 7 units of goods.

[24]We do not believe it is appropriate to compare the performance between the different aggregation operators. This is because the users' intention in choosing the operators reflects different objectives which are, in turn, reflected in the different semantics of the operators. The weighted average operator is used to balance all the evaluations; the weighted Einstein operator to satisfy all the evaluations more or less; and the weighted uninorm operator to compromise positive and negative evaluations.

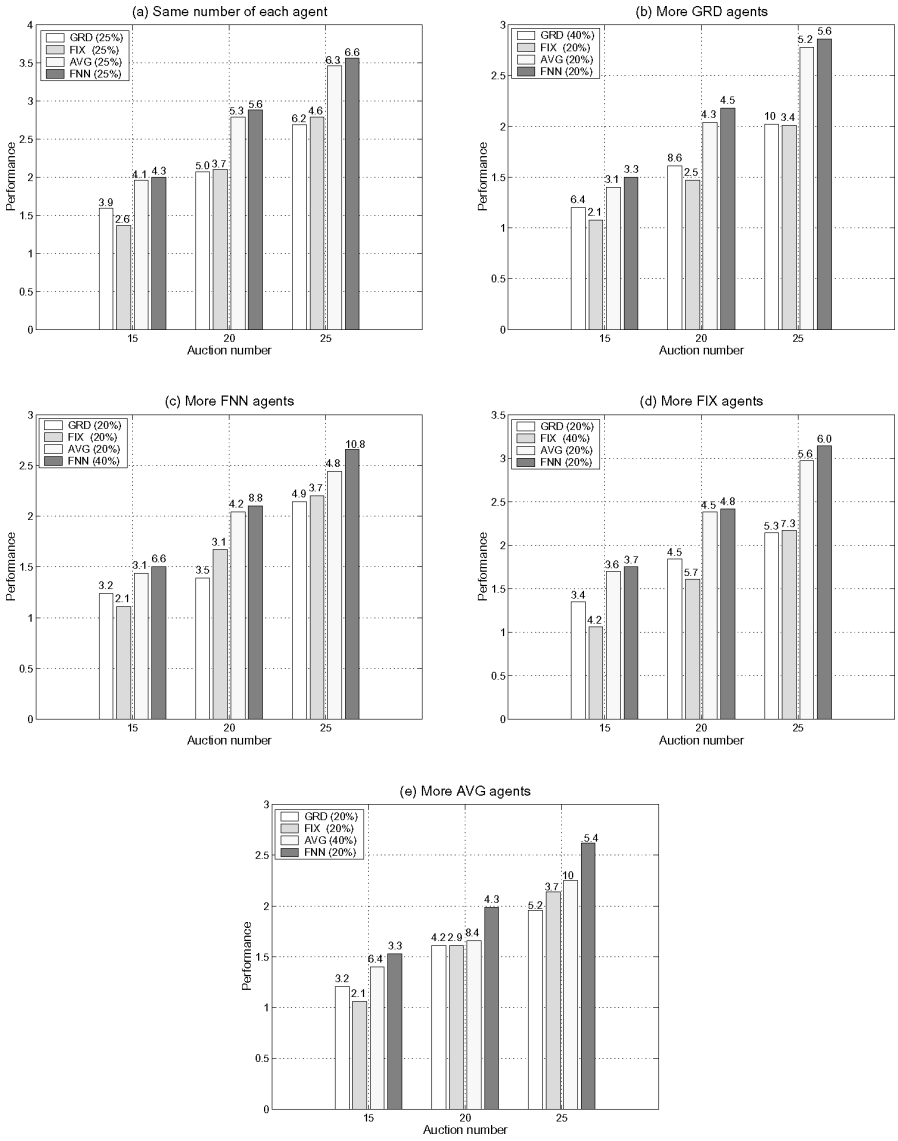

Fig. 13. Performance of agent with various agent populations. The horizontal axis shows the total auction number in the market. The vertical axis represents the performance of the kind of agents in that session. The number on top of each bar is the number of transactions made per agent by that kind of agents in the session. In (a), there are 4 agents in total, and 5 for (b) to (e). The demand for (a) is 32 and 35 for (b) to (e).

operator (see Section 5.3 for experiments with differently weighted attributes). This time, the numbers of each agent type in a given game are randomly generated.

As can be seen, the FNN agents behave the best, and AVG agents behave second in all cases. In fact, the order of performance of the four kinds of agent does

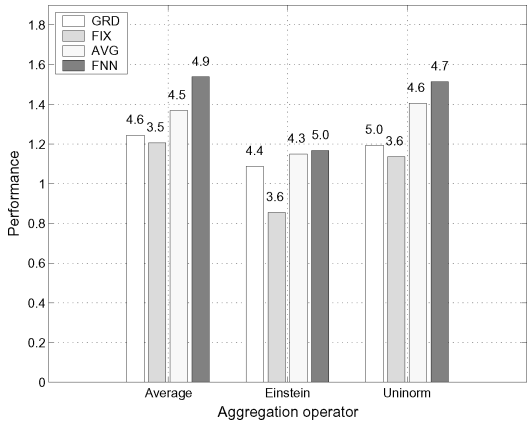

Fig. 14. Performance of various agents using different aggregation operators.

not change for different aggregators. This shows that our FNN agent performs best whatever aggregation operator is used. Again, we attribute this success of FNN agents to the efficiency of the ECF algorithm and its ability to predict the auction closing price and the parameter adaptation through learning.

5.3 Varying Preference Weights

This experiment evaluates the performance of the different agents when they use varying weights for the different attributes of the goods. This is an important issue to consider because, again, various weightings may lead to a different ranking of the strategies. Thus, for each operator, we conducted experiments to test the impact of the weights on the performance of various strategies. In this case, the number of auctions is generated randomly from $[15, 25]$ and Figure 15 shows the performance of the agents with the three representative weights we consider. As can be seen, the order of each kind of agent does not change for the different weight ratios. Again, in all cases, FNN agents perform the best, followed by AVG agents. This superior performance is due to the efficiency of the ECF algorithm.

However, in Figure 15(a) and (b), we see that the GRD agents also perform well when $w_p : w_q = 1 : 3$. This is because such agents always choose the currently best candidate, and the one with a high satisfaction on date is usually chosen. Note that GRD agents accept any price within their budget line, thus the satisfaction degree on price is not always high. Thus, when date is valued more, GRD agents tend to perform well. However, this advantage disappears when price is valued more. This feature is less apparent when we use the uninorm operator (see Figure 15(c)) since even when the satisfaction for one attribute is high, the overall evaluation is not always high.

5.4 Varying Attitudes to Risk

This experiment aims to study the influence of the risk attitude on the performance of the FNN agent. Specifically, we aim to find the best configuration of risk attitude for an agent to adopt. Thus, for each kind of aggregation

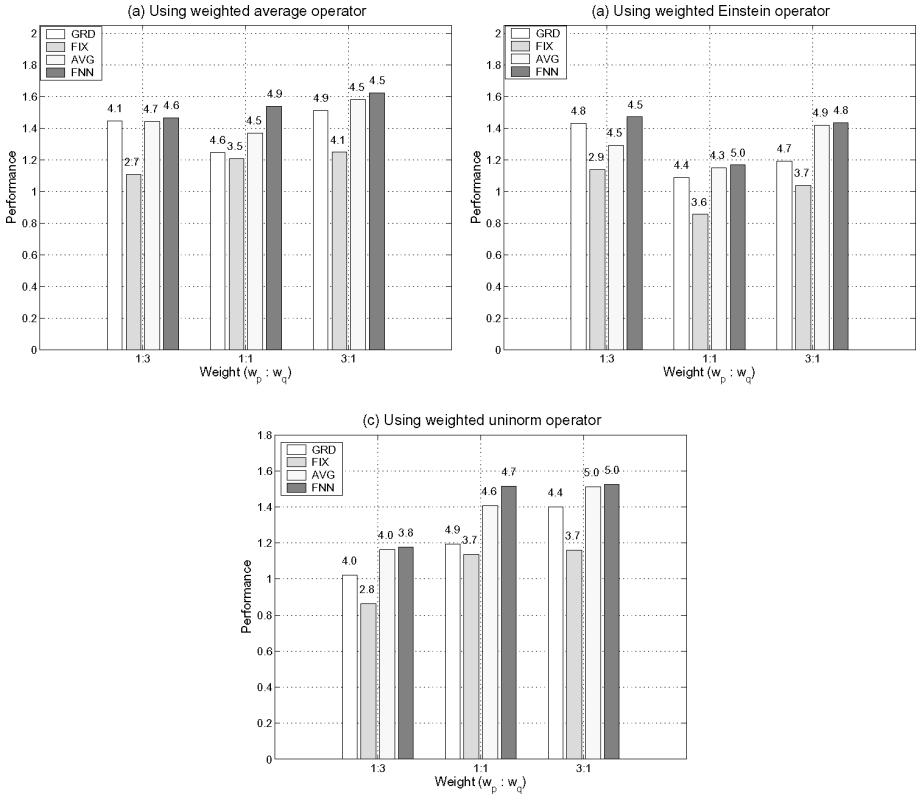

Fig. 15. Using weighted operators with varying weights.

operator, three risk attitudes are compared: risk-seeking, risk-neutral, and risk-averse. For example, when the weighted average operator is used, the threshold (λ in Figure 1) is 0.1 for a risk-seeking agent, 0.2 for a risk-neutral agent, and 0.3 for a risk-averse agent. From Figure 16, we can see that, in general terms, the higher the supply there is, the better the agent performs (as was shown in the experiment in Figure 13). However, the general trend is that risk-seeking agents perform better when supply is high, and a risk-averse agent behaves better when supply is low. This is because when supply is high, there is little competition among the agents, and a risk-seeking agent can win in the auction it selects easily, while when supply is low, the game is very competitive and a risk-averse agent can win a good whenever it is within the threshold.

6. RELATED WORK

There are three strands of work that are directly related to what we have described in this article. First, there is work on agents bidding in multiple overlapping auctions. Second, there is work on multi-attribute auctions. Third, there is work using fuzzy techniques to manage agent interactions. Each of these is now dealt with in turn.

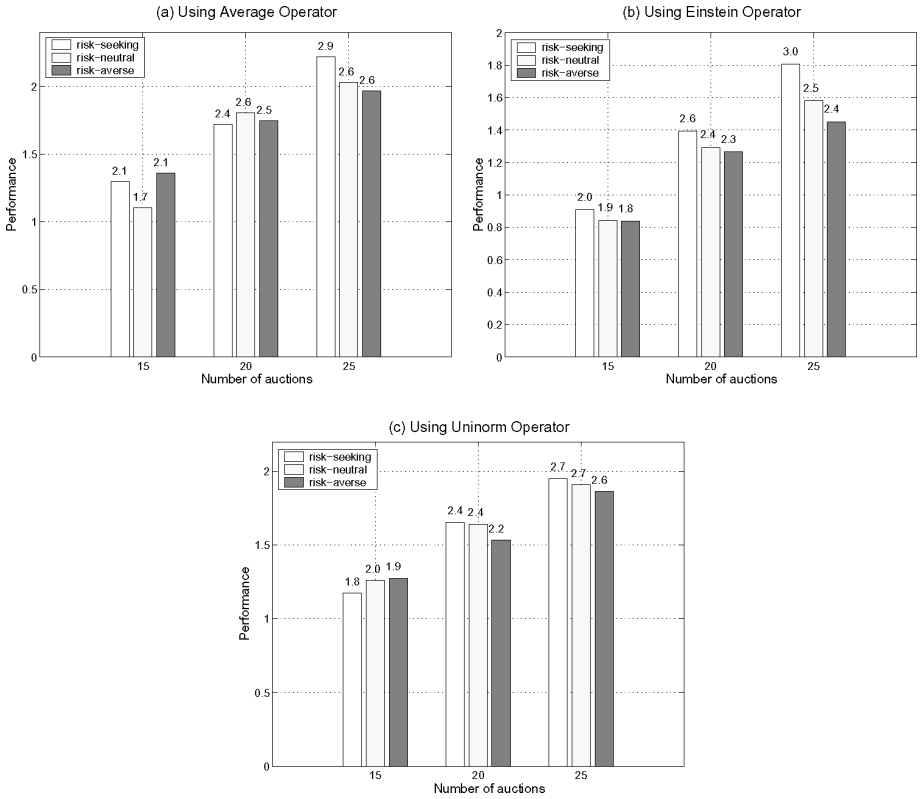

Fig. 16. Performance of FNN agents with different risk attitudes.

First, we consider bidding in multiple overlapping auctions. In this area, Preist designed an algorithm for agents that participate in multiple English auctions [Preist et al. 2001]. His algorithm proposes a coordination mechanism that can be used in cases where all the auctions terminate simultaneously and a learning method that uses a belief function about the valuations of bidders to tackle auctions that terminate at different times. It uses the belief function and the payoff (provided the good is obtained) to calculate the expected utility of a bid. However, an important shortcoming of this method is that it only considers the transaction price and bids in the history of the auction house but ignores the current state of the running auctions. Our ECF algorithm considers both factors. Moreover, his algorithm is only evaluated for cases where the auctions completely overlap. Thus it is not clear how it would perform for cases where there are varying start and end times. In a similar vein, Byde describes a dynamic programming approach for agents that participate in multiple English auctions to buy a single item [Byde 2001b]. Moreover, in Byde [2001a], the dynamic programming approach is compared with other algorithms and shown to be effective in obtaining high utilities. However, we believe it is difficult to extend this approach to a time-constrained environment (like many e-commerce scenarios) because of the heavy computational demands of this

technique. Building on this initial work, Byde et al. [2002] developed a framework that enables an agent to make rational bidding decisions across multiple heterogeneous auctions (English, Dutch, First-price sealed-bid and Vickrey auctions) with varying start and end times. It does this by using a fixed-auction strategy (like the one we use in Section 5) and a fixed threshold strategy to estimate the expected utility of a bid in a probabilistic way. However, in many cases, we believe it will prove difficult to obtain the probability values required for this model (for example, the probability that a given price will win an auction, and the probability that the highest opposing bid in the auction has a value less than a given price). Our approach circumvents this issue by using the fuzzy neural network to estimate the auction's closing prices, and then selecting the auction to bid in. Thus what we need for this reasoning model is some training patterns collected from previous games, and we believe this information is easier to obtain than the aforementioned probability distribution functions. Anthony and Jennings [2002] 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 the remaining time the number of remaining auctions, the desire for bargain, and the desperateness of the agent. The overall strategy is to combine these four tactics using a set of relative weights provided by the user. In an extension to this model [Anthony and Jennings 2003], a genetic algorithm is used to search for effective strategies so that an agent can behave appropriately according to its assessment of its prevailing circumstances. However, this method does not calculate the expected closing price of the various auctions. This means the agent has no well-founded basis for calculating the maximum bid it can offer at any moment in time. In this sense, it is like the Greedy strategy, but the agent will not accept a transaction with a low utility. However, as our experiments highlight, such a strategy is unlikely to maximize its satisfaction degree because it often misses out on better deals that may occur in the future.

All the strategies discussed involve price only in the negotiation. There is, however, some work that involves trading with multi-attribute goods. In particular, the Trading Agent Competition (TAC) (http://www.sics.se/tac) provides a platform for designers to develop software agents that can compete with one another in multiple auctions for complimentary and substitutable goods. Each participating agent simulates a travel agent with the goal of assembling a number of travel packages (flight tickets, hotel rooms, and entertainment tickets) for its eight customers. The key features of TAC are that there are 28 auctions of three different types (one-seller English auctions; continuous double auction, and 16th price English auctions) and the goods are related (for example, the customer stays in the same hotel for every night, and the outflight date should be after the inflight data, etc). The objective of an agent is to maximize the total satisfaction of its customers (that is, the sum of the customers' utilities). In TAC, although there are some similarities with our scenario, there are also important differences. In particular, the TAC involves multiple types of auctions with interdependent goods. Moreover, their English auctions are in a different form than those used in our context. Thus the strategies tend to be specific to this competition context and cannot easily be used in more

general settings. Interestingly, however, Stone et al. [2003] designed a model of the empirical price dynamics for this problem based on past data, and they then use the model to analytically calculate the optimal bids. Moreover, Wellman et al. [2004] surveyed various prediction techniques which were employed by the TAC entrants. These approaches include historical averaging, machine learning, and competitive analysis.

In a related area, Che [1993] investigates government procurement using a two-dimensional auction (price and quality). In his work, a buyer solicits bids from multiple sellers. Each bidder submits a sealed bid specifying the price and quality, and the bidder with the highest score wins. Based on the different ways in which the winner offers the goods/services, three auction schemes are proposed: first score (winner offers the price and quality it bids), second score (winner offers the goods/services matching the score of the second highest scored bidder), and second preferred offer (winner offers the goods/services at the same price and quality as the second highest scored bidder). In this model, the buyer evaluates the bids by a scoring function which converts a bid into a single number. Other work on multi-attribute auctions includes the design of a bidding procedure [Bichler et al. 1999], a multi-attribute auction protocol for service allocation [Jennings et al. 2000], and the English auction protocol for multi-attribute items [David et al. 2002]. However, in all cases, the difference with our work is that the agent bids on price and quality, while our agent only bids on price in the selected auctions where the other attributes are acceptable.

We now turn to the use of fuzzy techniques to manage an agent's interactions. In this vein, Faratin et al. [2002] used fuzzy similarity to compute trade-offs among multiple attributes during bilateral negotiations. Here fuzzy techniques are used to deal with a bilateral negotiation, and the algorithm aims to find a win-win (cooperative) solution for both parties. Luo et. al [2003] developed a fuzzy constraint-based framework for bilateral multi-issue negotiations in semicompetitive trading environments. Fuzzy sets are used to express users' preferences (as in our model), and a fuzzy aggregation method is used to evaluate an offer. In previous work [He et al. 2003], we developed a fuzzy logic-based bidding strategy for a continuous double auction and showed it outperformed a number of the standard strategies for this problem. Our approach uses a fuzzy reasoning technique to generate the bid or ask the agent can submit. But, as discussed earlier, the parameter adaptation is somewhat limited. When taken together, these models show that fuzzy techniques appear well suited to dealing with complex negotiation and bidding strategies. Their success occurs because fuzzy sets provide a natural way to flexibly represent the terms for preferences (for example, about 15 or less than 300) or factors involved in making bidding decisions (for example, price is high or number of auctions is small) that involve a high degree of uncertainty.

7. CONCLUSIONS AND FUTURE WORK

This article developed a new algorithm that guides an agent's bidding behavior in multiple overlapping English auctions for multiple items characterized by multiple attributes. The Earliest Closest First algorithm we developed first

calculates the auctions that best fit the users' preferences, and then bids in order of increasing end time in any auctions that have a satisfaction degree that is reasonably close to what are predicted to be the best ones. Specifically, the FNN implementation of this strategy uses neurofuzzy 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 decision making criteria of our agent to be adapted to the situation in which it finds itself. Specifically, the adaptation is based on the learning of the neural network where the parameters in the fuzzy sets and the consequent output can be adjusted. Moreover, we benchmarked our algorithm against two common alternatives available in the literature and the strategy which also uses ECF but with a different prediction function. In all the cases we considered, the FNN strategy performs the best. This shows the effectiveness of the ECF and the adaptation ability of the FNN implementation. Our algorithm can also make trade-offs in its bidding behavior between the different attributes that characterize the desired good in order to maximize the user's satisfaction.

For the future, there are five main extensions required for our model. First, the algorithm can be extended to consider multiple attributes with interdependent attributes (for example, the hotel rooms are useful only after the day of arrival). In this case, the fuzzy rule base will need to be expanded to deal with the relation among attributes. Second, the FNN structure of the FNN agent (for example, size of the linguistic terms and the fuzzy rules) is currently designed by domain experts. Ideally, however, this structure should be obtained by self-organized learning [Lin 1994] so users can employ it more readily. Third, the idea of ECF can be extended to other auction protocols (for example, Continuous Double or Dutch auctions). In this case, we believe it is likely that the best asks and bids can be calculated first and then according to the current ask or bids, the agent can decide whether to accept the ask/bid or submit a fuzzified ask/bid. Fourth, it would be desirable for the risk attitude parameter to be self-tuned. Thus it could be adjusted according to how frequently the agent has transaction or how eager it is to trade. Fifth, it is interesting to compare our FNN prediction approach with other prediction techniques such as the Bayesian networks or other form of neural network in this scenario. Inspired by MacKie-Mason et al. [2004], and Walsh et al. [2002], which uses a game theoretic approach to find the dominant strategy or the most plausible equilibria, we intend to do some similar experiments to compare the performance of our FNN agents with those that are regular neural networks and Bayesian networks.

REFERENCES

ANTHONY, P. AND JENNINGS, N. R. 2002. Evolving bidding strategies for multiple auctions. In *Proceedings of the 15th European conference on Artificial Intelligence*, F. van Harmelen, Ed. IOS Press, Amsterdam, The Netherlands, 178–182.

ANTHONY, P. AND JENNINGS, N. R. 2003. Developing a bidding agent for multiple heterogeneous auctions. *ACM Trans. on Internet Techn. 3*, 3, 185–217.

BICHLER, M., KAUKAL, M., AND SEGEV, A. 1999. Multi-attribute auctions for electronic procurement. In *Proceedings of the 1st IBM IAC Workshop on Internet Based Negotiation Technologies*. Yorktown Heights, NY.

BYDE, A. 2001a. A comparison among bidding algorithms for multiple auctions. Tech. rep. HP Research Labs.

BYDE, A. 2001b. An optimal dynamic programming model for algorithm design in simultaneous auctions. Tech. Rep. HPL-2001-67, (March). Hewlett Packard Research Labs, Bristol, UK.

BYDE, A., PREIST, C., AND JENNINGS, N. R. 2002. Decision procedures for multiple auctions. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems*. 613–620.

CHE, Y.-K. 1993. Design competition through multidimensional auctions. *RAND J. Economics 24*, 4, 668–680.

DAVID, E., AZOULAY-SCHWARTZ, R., AND KRAUS, S. 2002. An english auction protocol for multi-attributes items. In *Agent Mediated Electronic Commerce*, O. Shehory and W. Walsh, Eds. Lecture Notes in Artificial Intelligence, vol. 2531. Springer, 52–68.

FARATIN, P., SIERRA, C., AND JENNINGS, N. R. 2002. Using similarity criteria to make trade-offs in automated negotiations. *Artificial Intelligence 142*, 2, 205–237.

FLETCHER, R. AND REEVES, C. 1964. Function minimization by conjugate gradients. *Comput. J. 7*, 2, 149–154.

HE, M. AND JENNINGS, N. R. 2004. Designing a successful trading agent: A fuzzy set approach. *IEEE Trans. Fuzzy Syst. 12*, 3, 389–410.

HE, M., JENNINGS, N. R., AND LEUNG, H. F. 2003. On agent-mediated electronic commerce. *IEEE Trans. Knowl. Data Engin. 15*, 4, 985–1003.

HE, M., LEUNG, H. F., AND JENNINGS, N. R. 2003. A fuzzy logic based bidding strategy for autonomous agents in continuous double auctions. *IEEE Trans. Knowl. Data Engin. 15*, 6, 1345–1363.

JANG, J. 1993. ANFIS: adaptive-network-based fuzzy inference systems. *IEEE Trans. Syst., Man and Cybernet. 23*, 3, 665–685.

JENNINGS, N. R. 2001. An agent-based approach for building complex software systems. *Comm. ACM 44*, 4, 35–41.

JENNINGS, N. R., NORMAN, T., FARATIN, P., O'BRIEN, P. AND ODGERS, B. 2000. Autonomous agents for business process management. *Appl. Artificial Intelligence 14*, 2, 145–189.

JOHANSSON, E., DOWLA, F., AND GOODMAN, D. 1990. *Back-Propagation Learning for Multi-Layer Feed Forward Neural Networks Using the Conjugate Gradient Method*. Lawrence Livermore National Laboratory, CA.

JONKER, R. AND VOLGENANT, A. 1987. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Comput. 38*, 325–340.

KEPHART, J. 2002. Software agents and the route to the information economy. *Proceedings of the National Academy of Sciences 99*, 7207–7213.

LIN, C. 1994. *Neural Fuzzy Control Systems with Structure and Parameter Learning*. World Scientific.

LUCKING-REILEY, D. 2000. Auctions on the Internet: What's being auctioned, and how? *J. Industr. Economics 48*, 3, 227–252.

LUO, X., JENNINGS, N. R., SHADBOLT, N., LEUNG, H., AND LEE, J. 2003. A fuzzy constraint based model for bilateral, multi-issue negotiation in semi-competitive environments. *Artificial Intelligence 148*, 1–2, 53–102.

LUO, X., LEE, J., LEUNG, H., AND JENNINGS, N. R. 2003. Prioritised fuzzy constraint satisfaction problems: Axioms, instantiation and validation. *Fuzzy Sets syst. 136*, 151–188.

MACKIE-MASON, J., OSEPAYSHVILI, A., REEVES, D., AND WELLMAN, M. 2004. Price prediction strategies for market-based scheduling. In *the 14th International Conference on Automated Planning and Scheduling*.

PREIST, C., BYDE, A., AND BARTOLINI, C. 2001. Economic dynamics of agents in multiple autions. In *Proceedings of the 5th International Conference on Autonomous Agents*. Montreal, Canada, 545–551.

RUMELHART, D., HINTON, G., AND WILLIAMS, R. 1986. Learning internal represenations by error propagation. *Parall. Distribu. Process. 1*, 318–362.

RUST, J., MILLER, J., AND PALMER, R. 1991. Behavior of trading automata in a computerized double auction market. In *The Double Auction Market: Institutions, Theories, and Evidence*, D. Friedman and J. Rust, Eds. Addison-Wesley, 155–198.

STONE, P., SCHAPIRE, R., LITTMAN, M. L., CSIRIK, J. A., AND MCALLESTERA, D. 2003. Decision-theoretic bidding based on learned density models in simultaneous, interacting auctions. *J. Artificial Intelligence Resear. 19*, 209–242.

TSVETOVAT, M. AND SYCARA, K. 2000. Customer coalitions in the electronic marketplace. In *Proceedings of the 4th International Conference on Autonomous Agents*. Spain, 263–264.

WALSH, W., DAS, R., TESAURO, G., AND KEPHARI, J. 2002. Analyzing complex strategic interactions in multi-agent games. In *AAAI-02 Workshop on Game Theoretic and Decision Theoretic Agents*. 109–118.

WELLMAN, M., REEVES, D., LOCHNER, K., AND VOROBEYCHIK, Y. 2004. Price prediction in a trading agent competition. *J. Artificial Intelligence Resear. 21*, 19–36.

WURMAN, P., WELLMAN, M., AND WALSH, W. 2001. A parametrization of the auction design space. *Games Economic Behavior 35*, 304–338.

YAGER, R. 1994. Aggregation operators and fuzzy systems modeling. *Fuzzy Sets Syst. 67*, 129–145.

YAGER, R. AND RYBALOV, A. 1996. Uninorm aggregation operators. *Fuzzy Sets and Syst. 80*, 111–120.

ZADEH, L. 1965. Fuzzy sets. *Inform. Control 8*, 338–353.

ZADEH, L. 1975. The calculus of fuzzy restrictions. In *Fuzzy Sets and Applications to Cognitive and Decision Making Processes*, L. Z. et. al., Ed. Academic Press, 1–39.

Received January 2005; accepted June 2005