# Closing Price Prediction for Auctions of Hotel Rooms on the TAC Classic

Eber Jair Flores Andonegui[1], Darnes Vilariño Ayala[1], Fabiola López y López[2]

[1] Facultad de Ciencias de la Computación, BUAP
Puebla, México
[2] Dirección de Modalidades Alternativas de Educación, BUAP
Puebla, México
eberjair@hotmail.com darnes@cs.buap.mx, fabiola.lopez@siu.buap.mx

*(Paper received on June 20, 2007, accepted on September 1, 2007)*

**Abstract.** In this paper a brief description of the Trading Agent Competition (TAC) Classic and the problems that agents must face are presented. Moreover, episodes are proposed as a solution to predict the closing price of hotel rooms for a bid in the competition. To do that, a five layer neural network is used and, after upgrading the original neural network twice, the results, obtained from 20 games among dummy and other agents are shown.

## 1 Introduction

The Trading Agent Competition (TAC) has been organized during many years ago. In this competition, travel agents compete against each other in order to better satisfy the requirements of their customers. What makes the game interesting is the fact that every trading agent is a software agent and, it is specialized on selling travel packages to determined customers. Each travel package includes flights and hotel room reservations as well as tickets for entertainment activities of the customers [1].

The main problem is that every travel agent must propose to its customer the best day to travel, according to the conditions of the market, the customer requirements on hotel quality and desired entertainment and, flights availability. Travel agents work in an autonomous way, i.e. they must be able to take decisions on their own. To do so, an architecture that includes the following components has been proposed: (a) learning schemas that allow agents to adapt their behavior to the current environment, (b) mechanisms to interact with the platform and other participants and (c) capabilities to analyze the changing conditions on the market.

Many papers have been written about the TAC Classic, and some of the production of 2004 and 2005 is referred. For example, Sardinha et al 1 present a multiagent architecture that solves this problem. Here, each agent of the system specializes in a part of the market, and it can predict the purchase prices of tickets in the following next days by checking the room availability. This multiagent system considers a learning architecture that is supported on a knowledge base which is created during the game time. Pannos Toullis et al2 approach the problem by using fuzzy logia and reasoning rules to predict the room prices, by considering available historical data. Trying to efficiently assign the amusement shows, they design a strategy that guarantees the requirements of


Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 161-172

each client. Michael P. Wellman et al. on 3 proposes a hierarchical distribution of the original problem, by analyzing and predicting the behavior of a rival agent. Against one rival, the Walverine agent may predict the whole of its behavior. Against two rivals, the agent may predict a 74.1% of their behavior. But if the rivals are extended to four of them, the agent may predict only the 2.4% of their behavior. In general, those proposals consist on multi-agent systems that allow the specialization of each subagent on a part of the whole market.

Our research has been done in three different phases: an analysis of the conditions of the environment where agents participate; a study of the platform where the competition takes place; and the design and implementation of a Neural Network that can predict the closing bidding prices for the hotel rooms. The paper is structured as follows: Section 2 gives a brief description of the TAC Classic. Section 3 describes the main problems of the game. Section 4 shows the design of the neural network that can predict the closing bidding prices for the hotel rooms. Section 5 presents the results for the local tests over the neural network itself. Finally, Section 6 contains the conclusions about our results.

## 2 The TAC Classic Description

The TAC Classic provides a platform where simulations of games can be done. The platform is in charge of the following tasks: hosting the participants; activating the agents once the simulation starts and removing them once the simulation ends; and providing relevant information from each round. Some of the data obtained from the platform are used as configuration values to participate on further simulations. These data consist on the bidding for hotel rooms, flights and entertainment shows prices.

In each TAC Classic game, 8 travel agents participate. These agents organize travel tours from TACTown to Tampa for eight different customers during an imaginary period of 5 days. Customers express their preferences for every aspect of the trip, Their preferences are represented by a utility function [1]. The goal of each agent is to maximize the total utility of their customers. Agents compete against each other to obtain the best result on assembling the tours of their clients. They participate on 28 bids. A tour consists of a round flight, hotel reservation and tickets for entertainment events such as the alligator wrestling, an amusement park or a museum.

Each one of these goods –flights, hotel reservations and entertainment events- are traded in separate markets with different rules as following:

There are two hotels in Tampa: Tampa Towers (TT), which has a high price because it is the most comfortable hotel, and Shoreline Shanties (SS) which is the cheapest option for traveling to Tampa. Once the hotel has been assigned, it cannot be changed during the staying of the client in Tampa. Since the client needs hotel on his arrival night and until the pre-departure night, there is no hotel available on the last day of game. To get a room, the agent must participate on ascending auctions: one auction for each combination of hotel and night, each with 16 rooms being auctioned off. The price of each hotel is based on the customer preference for the Tampa Towers Hotel. The agents are not allowed to exchange hotel rooms between themselves. For a deeper

specification of the other parameters such as flight reservations and entertainment events, we recommend to check [1].

Regarding hotel rooms, each agent must send an offer for the auction. While it is open, the auction price is updated with the current values. All of the bids are presented as a pair (q, p) where q is the quantity, and p is the price offered by the agent. If q is less than zero, then it is indicated that the agent wants to 'sell' a quantity q of tickets on a base price p. If q is over zero, then the agent is interested in 'purchasing' certain quantity q of tickets by paying p. The client preferences are specified as follows: The day when they wish to arrive to Tampa (PA); the departing day from Tampa (PD); a prize for assigning the best hotel (HP) -this value is between $50 and $150-; and a prize value for assigning an entertainment type -on a range between $0 and $200 according to the customer preferences: AW for alligator wrestling, AP for amusement park, and MU for museum.

The tour is specified by using these variables: the arriving day (AA), the departure day (AD), and a binary value (0 or 1) that indicates whether the Tampa Towers was selected (TT?).

The tour is feasible if it contains the hotel reservations for each night between the arrival and departure dates. The shortest stay on Tampa is for one day, and obviously the customer will not stay at the hotel while flying back to TACTown. It is important to remark that the obtained prize for reserving the Tampa Towers on a travel is for the whole tour and not for each night spent in Tampa.

The winning agent is the one that obtains the highest score of the game. The score is obtained from the obtained utilities less the costs for each tour. The profit of each client is measured in dollars and is calculated by the following utility function:

$$u = 1000 - travel\_penalty + hotel\_bonus + fun\_bonus \quad (1)$$

$$\text{where: } travel\_penalty = 100*(|AA - PA| + |AD - PD|);$$

$$hotel\_bonus = TT? * HP; \text{ and}$$

$$fun\_bonus = AW?*AW + AP?*AP + MU?*MU.$$

The parameter *hotel_ bonus* can be $0 if the client is not assigned to the Tampa Towers (TT?=0) for staying. If he is assigned (TT?=1) the hotel price will take the HP value defined previously, and it is added to the general utility.

## 3 Problems to be solved by the Agent

Agents should be able to offer clients optimized tour packages that satisfy most of their needs. Therefore, it is important for agents to get a reservation at the Tampa Towers in such a way that the pair *arrival-departure* days are in accordance with the desired staying days. In addition, agents must also get the required entertainment events for the staying days in Tampa.

One way to solve the problem is by programming separate modules that solve a part of the whole problem (i.e. dividing the problem into sub-problems, see Figure 1). Then, the agent makes its own decisions by considering the solutions found by each module.

Each module has been designed to operate different kinds of transactions, such as the hotel rooms by using English auction and the flight prices by using a stochastic function. Once the agent predicts the next possible scenario, it consults the possible strategies and the tasks that it should perform to confront the stage, and finally it saves the results of applying the current strategy. Once the decisions are made, the offers are sent to the server. One of the most important problems during the game is to guarantee a room for the clients during their stay in Tampa. To solve this problem, an artificial neural network is proposed, such as He et al [5] did.

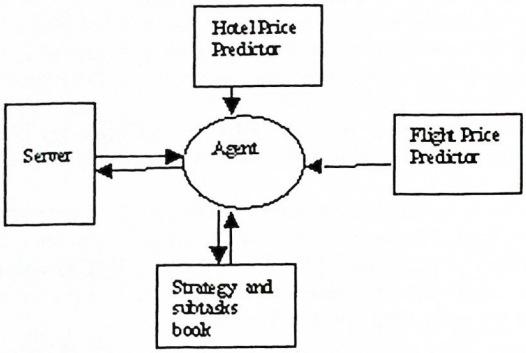


**Fig. 1.** Problem solution by programming separate modules.

To get the hotel rooms the agent must participate in auctions. The format of the auctions for the hotel rooms is English Standard, it means ascending from one to many. The only modification is that the auctions are closed randomly on every minute of game. Each auction is closed randomly during the eight minutes of the game. The agent does not know previously which auction will be closed during the next minute. Each hotel offers 16 rooms per night, on a minimum price of $0.

The requested price is announced every minute, and it is calculated by considering the sixteenth highest price of all the bids. When an agent makes a new bid, the suggested price must be higher than the one established, by considering the following rules: the offered price of a new bid must be at least one unit higher than the requested price. If the current offer acquires q rooms, the new offer must request at least q rooms, at least one unit higher than the requested price. Agents cannot retreat their bids. When an auction is closed, the 16 units with the highest offered prices will be sold and agents will pay the requested price. Since the requested price is known only every minute of the game, it is not sure that for an offered price, that the agent can obtain the required rooms. A possible solution can be found by trying to predict the requested price and make decisions about how much to spend for the required rooms.

## 4 Designing the Neural Network

The factors that are considered to predict the closing price of an auction are, first, the customer preference for the hotel type (*Pref*). A higher preference for the room tends to increase the offered price for it. *Pref* is fuzzily defined with the values *Low, Medium* and *High*. We also consider the average closing price on previous auctions for the same room (*Prom*). Some rooms tend to be more required, so the offering price could be increased. The fuzzy values for this variable are *Low, Medium* and *High*. Another factor that is considered is the distribution of the closed auctions (*Dist*). If the auctions of the hotel rooms on the closer days to the current auction are closed, the price for the same room may tend to increase because clients cannot change the hotel during their staying at Tampa. The fuzzy values considered are *Spread, Half* and *Gathered*. The only values this variable can take are (a) 0 if no auction from the two nearest auctions to the current day has closed; (b) 1 for one of the two auctions that has closed; and (c) 2 if both auctions are closed. For the expected price (*Price*) five fuzzy values were considered: *Very_Low, Low, Medium, High* and *Very_High*. We create 27 rules of fuzzy reasoning and a 5-layer neural network.

The values for the input variables correspond to the first layer. Each node in this layer returns a belonging value for the input. For the cases of the values of Pref and Prom, a normal distribution functions is given as follows:

$$S_i^{(1)} = e^{-\frac{(x-\bar{x}_i)^2}{2\delta_i^2}} \quad \text{for } i = 1\ldots6 \tag{2}$$

Where x is the input value and the parameters $\bar{x}$ and $\delta$ are fixed when the neural network learns from its environment in order to reduce the prediction error for all of the corresponding nodes to the variable *Pref*. For all of the nodes related to the variable *Prom*, the parameters $\bar{x}$ and $\delta$ are calculated as follows:
Let MB to be the lowest average price, and CS the highest average price, then:

$$\bar{x}_i = \begin{cases} \dfrac{CS+5MB}{6} & i=4 \\ \dfrac{CS+MB}{2} & i=5 \\ \dfrac{5CS+MB}{6} & i=6 \end{cases} \tag{3}$$

$$\delta_i = \begin{cases} \bar{x}_i - MB & i=4 \\ \bar{x}_i - \dfrac{CS+2MB}{3} & i=5 \\ CS-\bar{x}_i & i=6 \end{cases} \tag{4}$$

It can be observed that the assignment of values for these parameters depends on the general behavior of the auctions, and it does not depend on the neural network itself. It means that the assignment is not fixed through the learning process. To determine the factor *Dist*, a discrete activation function is used, it is given by:

$$S_i^{(1)} = \begin{cases} a_i & \text{if } x = 0 \\ b_i & \text{if } x = 1 \\ c_i & \text{if } x = 2 \end{cases} \quad \text{for } i = 7...9 \tag{5}$$

Where the values $a$, $b$ and $c \in (0,1]$, and they are fixed when the neural network learns to decrease the prediction error. The second layer belongs to the inference rules. This layer determines the strength of the rule by multiplying all of the inputs:

$$S_i^{(2)} = p_i = \prod_{j \in C_i^{(1)}} S_j^{(1)} \text{ for } i = 1...27 \tag{6}$$

Where $C_i^{(1)}$ is the group of nodes from the first layer, that pass information to the $i$ node on the second layer. The third layer calculates the importance degree for each rule, or in other words, the relative strength of each rule by following the function:

$$S_i^{(3)} = p'_i = \frac{p_i}{\sum_{j \in C^{(2)}} p_j} \tag{7}$$

Where $C^{(2)}$ is the group of nodes from the second layer that correspond to the $i$ node. The fourth layer accumulates the weights of each rule, by considering the same output, to generate the value for the output variable, by applying the following function:

$$S_i^{(4)} = r_i \sum_{j \in C_i^{(3)}} p'_j \tag{8}$$

Where $C_i^{(3)}$ is the group of nodes from the previous layer that corresponds to the $i$ node of the current layer. Finally the fifth layer gathers all the values of the fuzzy output variables from the fourth layer, with their assigned weight as follows:

$$S^{(5)} = \sum_{i \in C^{(4)}} (r_i \sum_{j \in C_i^{(3)}} p'_j) \tag{9}$$

Where $C^{(4)}$ is the group of nodes from the fourth layer. Once the price is obtained on the output of the neural network, if it is smaller or equal to the requested price in the current auction, then the requested price is taken, and increased over a 10%, as the next offering price. A second version of the same neural network can be obtained by considering the current asked price *Asked_Price* as an input on the first layer, instead of the customer preference *Pref*. Using the same fuzzy rules the functioning of the new network is explained as follows: A current asked price *Asked_Price* means that some agents are pursuing the same auction, and then it tends to increase the next requesting price. During the training of the neural network the parameters are adjusted, in order to guarantee the learning of the network itself. The main objective is to minimize the error value, which is calculated by:

$$E = \sum_j \frac{1}{2} (Y_j - S_j^{(5)})^2 \tag{10}$$

Every time that a new group of data is received, the output values of all the nodes are saved, in order to calculate $\frac{\partial E}{\partial S}$ of each node. These values are considered on the application of the optimizing algorithms to minimize the errors on the calculations. The first adjusted parameters are the $r_i$ parameters that are on the fourth layer. The applied technique, as applied on the paper of He et al. [5] is based on the optimal direction of maximal decrease:

$$r(t+1) = r(t) + \eta(-\frac{\partial E}{\partial r}) \tag{11}$$

Where $\eta$ is the learning factor, that is initially considered as $\eta = .01$. The inference rules to adjust the parameters $r_i$ in the fourth layer is:

$$\frac{\partial E}{\partial r_i} = \frac{\partial E}{\partial S^{(5)}} \frac{\partial S^{(5)}}{\partial S_i^{(4)}} \frac{\partial S_i^{(4)}}{\partial r_i} = (S^{(5)} - Y) \sum_{j \in C_i^{(3)}} p'_j \tag{12}$$

Then, $r_i$ is updated as follows:

$$r_i(t+1) = r_i(t) - \eta(S^{(5)} - Y) \sum_{j \in C_i^{(3)}} p'_j \tag{13}$$

The following parameters that must be adjusted are $\bar{x}$ and $\delta$ for all the nodes of the first layer that correspond to the input variable *Pref*. The implemented technique is based on the optimization by targeted gradients. Let it be $g_t$ the gradient of the iteration $t$ ($t$>1), then the new search direction is obtained by combining the current maximum descent direction against the previous direction, that is:

$$p_t = -g_t + \beta_t p_{t-1} \tag{14}$$

By using Fletcher-Reeves's Upgrade it is obtained:

$$\beta_t = \frac{g_t^T g_t}{g_{t-1}^T g_{t-1}} \tag{15}$$

Then for modifying the learning rules for the parameter $\bar{x}$, a gradient that considers the next partial derivative as the error function –where $C_{-i}^{(2)}$ is the group of nodes from the second layer that are connected with the $i$ node of the first layer as follows:

$$\frac{\partial E}{\partial \bar{x}_i} = \sum_{m \in C_{-i}^{(2)}} \left( \frac{\partial E}{\partial S_m^{(2)}} \frac{\partial S_m^{(2)}}{\partial S_i^{(1)}} \frac{\partial S_i^{(1)}}{\partial \bar{x}_i} \right) \tag{16}$$

$$\frac{\partial E}{\partial \bar{x}_i} = \sum_{m \in C_{-i}^{(2)}} \left( \sum_{k \in S_{-m}^{(3)}} \left( \frac{\partial E}{\partial S_k^{(3)}} \frac{\partial S_k^{(3)}}{\partial S_m^{(2)}} \right) \frac{\partial S_m^{(2)}}{\partial S_i^{(1)}} \frac{\partial S_i^{(1)}}{\partial \bar{x}_i} \right) \tag{17}$$

where:

$$\frac{\partial E}{\partial S_k^{(3)}} = \left(S^{(5)} - Y\right)_k; \quad \frac{\partial S_k^{(3)}}{\partial S_m^{(2)}} = \begin{cases} \dfrac{\sum_k p_k - p_m}{\left(\sum_k p_k\right)^2} & do\ if\ k = m, \\ \dfrac{-p_m}{\left(\sum_k p_k\right)^2} & do\ if\ k \neq m, \end{cases}$$

$$\frac{\partial S_m^{(2)}}{\partial S_i^{(1)}} = \frac{p_m}{S_i^{(1)}}; \text{ and } \frac{\partial S_i^{(1)}}{\partial \bar{x}_i} = e^{-\frac{(x_i - \bar{x}_i)^2}{2\delta_i^2}} \frac{\left(x_i - \bar{x}_i\right)}{\delta_i^2}$$

The rule for adjusting the parameter $\bar{x}_i$ is:

$$\bar{x}_i(t+1) = \bar{x}_i(t) + \eta p_t \tag{18}$$

where

$$p_t = -g_t + \beta_t p_{t-1}; \text{ and } g_t = \frac{\partial E}{\partial \bar{x}_i}$$

Similarly for $\delta_i$ the adjusting rule is calculated from:

$$\frac{\partial E}{\partial \delta_i} = \sum_{m \in C_{-i}^{(2)}} \left( \sum_{k \in S_{-m}^{(3)}} \left( \frac{\partial E}{\partial S_k^{(3)}} \frac{\partial S_k^{(3)}}{\partial S_m^{(2)}} \right) \frac{\partial S_m^{(2)}}{\partial S_i^{(1)}} \frac{\partial S_i^{(1)}}{\partial \delta_i} \right) \tag{19}$$

where

$$\frac{\partial S_i^{(1)}}{\partial \delta_i} = e^{-\frac{(x_i - \bar{x}_i)^2}{2\delta_i^2}} \frac{(x_i - \bar{x}_i)^2}{\delta_i^3}.$$

The rule for fixing the parameter $\delta_i$ is:

$$\delta_i(t+1) = \delta_i(t) + \eta p_t \tag{20}$$

where

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

For the nodes of the first layer that correspond to the variable Dist, the parameters $a$, $b$ and $c$ must be adjusted by using the same technique of conjugated gradients. Only one difference must be applied because a discrete activation function must be applied, so to find a direction to reduce the error, the Lagrange Interpolation is proposed. For this particular problem it is used:

$$P(x_i) = \frac{(x_i - 1)(x_i - 2)}{2} a - (x_i)(x_i - 2)b + \frac{(x_i)(x_i - 1)}{2} c \tag{21}$$

Once the polynomial is obtained, the adjusting rule for the parameter $a$ can be calculated, as on the parameters $\bar{x}$ and $\delta$ from the formulae:

$$\frac{\partial E}{\partial a_i} = \sum_{m \in C_{-i}^{(2)}} \left( \sum_{k \in S_{-m}^{(3)}} \left( \frac{\partial E}{\partial S_k^{(3)}} \frac{\partial S_k^{(3)}}{\partial S_m^{(2)}} \right) \frac{\partial S_m^{(2)}}{\partial S_i^{(1)}} \frac{\partial S_i^{(1)}}{\partial a_i} \right) \tag{22}$$

where

$$\frac{\partial S_i^{(1)}}{\partial a_i} = \frac{(x_i - 1)(x_i - 2)}{2}.$$

The adjusting rule for the parameter $a_i$ can be defined as follows:

$$a_i(t+1) = a_i(t) + \eta p_t \tag{23}$$

where

$$p_t = -g_t + \beta_t p_{t-1}; \quad \text{and} \quad g_t = \frac{\partial E}{\partial a_i}$$

Similarly, the adjusting rule for the $b$ parameter is calculated from the formulae:

$$\frac{\partial E}{\partial b_i} = \sum_{m \in C_{-i}^{(2)}} \left( \sum_{k \in S_{-m}^{(3)}} \left( \frac{\partial E}{\partial S_k^{(3)}} \frac{\partial S_k^{(3)}}{\partial S_m^{(2)}} \right) \frac{\partial S_m^{(2)}}{\partial S_i^{(1)}} \frac{\partial S_i^{(1)}}{\partial b_i} \right) \tag{24}$$

where

$$\frac{\partial S_i^{(1)}}{\partial b_i} = -(x_i)(x_i - 2) \cdot$$

And the adjusting rule for the parameter $b_i$ is defined by:

$$b_i(t+1) = b_i(t) + \eta p_t \tag{25}$$

where

$$p_t = -g_t + \beta_t p_{t-1}; \quad \text{and} \quad g_t = \frac{\partial E}{\partial b_i}$$

Finally the adjusting rule for the parameter $c$ is calculated by:

$$\frac{\partial E}{\partial c_i} = \sum_{m \in C_i^{(2)}} \left( \sum_{k \in S_m^{(3)}} \left( \frac{\partial E}{\partial S_k^{(3)}} \frac{\partial S_k^{(3)}}{\partial S_m^{(2)}} \right) \frac{\partial S_m^{(2)}}{\partial S_i^{(1)}} \frac{\partial S_i^{(1)}}{\partial c_i} \right) \tag{26}$$

where

$$\frac{\partial S_i^{(1)}}{\partial c_i} = \frac{(x_i)(x_i - 1)}{2}.$$

Then the adjusting rule for the parameter $c_i$ comes from:

$$c_i(t+1) = c_i(t) + \eta p_t \tag{27}$$

where

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

It can be observed from the characteristics of the interpolation and because it is a discrete function, that the error differential caused by the parameter variance is many times zero. This value brings itself a problem with the update value, because it becomes undetermined. When this happens, an updating value is proposed by default, $\beta = 1$. Due to this situation, a third version for the neural network is proposed. This version includes the updates made from the second version and some modifications of the calculation of the values for $a$, $b$ and $c$. This is made by implementing the maximum descent method, and it is indicated as follows:

$$a(t+1) = a(t) + \eta(-\frac{\partial E}{\partial a}) \tag{28}$$

$$b(t+1) = b(t) + \eta(-\frac{\partial E}{\partial b}) \tag{29}$$

$$c(t+1) = c(t) + \eta(-\frac{\partial E}{\partial c}) \tag{30}$$

Where, the differentials are calculated by using the same formulas from the first and second version. In the next section, the obtained results from the performance of the three versions of the neural network are presented.

## 5 Obtained Results

Many tests were developed in order to find the most efficient of the three versions of the neural network. The tests were made on a local server by simulating 20 games of the TAC Classic, for both learning factors 0.01 and 0.001, with the three implemented

versions of the neural network. The main objective is to find the best behavior for the prediction of the hotel room prices. To make the tests, two agents from the TAC Agent Repository [6] were downloaded and installed to compete in the platform –UTTA06 and MerTACor. The agents and the server were executed on the local network.

The three described versions of the neural network were tested, each with different input variables and optimization techniques. The networks RN1 and RN2 have the three input values –Preference, Average Price and Distribution-, and uses Conjugated Gradient for optimization and learning. For the third network RN3, only the Distribution input values were tested with Maximal Descent Optimization technique.

By using the learning factor f = 0.01 with RN1, the average prediction error is calculated in 49.5024696, with RN2 it is 56.233856, and with RN3 the error was 53.3619775. Considering a learning factor f = 0.001 the Average Prediction Error for RN1 is 59.7812932, with RN2, 52.07654 and with RN3, it is 52.0490384.

## 6 Conclusions

In this paper, the output and input values for the problem of the prediction of the hotel room prices were defined. Twenty-seven rules were established in order to build a neural network of five layers. The learning techniques implemented on the neural network, by using discrete activation functions was also explained, and is considered as the real contribution of this paper.

By changing the controlled variables, three different versions of the neural network were developed, named RN1, RN2 and RN3. By analyzing the obtained results on the presented tests, it can be inferred that RN3 with a learning factor f = 0.001 offers the best behavior of all the implementations. Then, an agent with this network will be used to predict the prices on the room hotel auctions.

A heuristic to solve the assignation problem for bidding prices for tickets for the amusement shows has been designed, and for the assignment of the remaining tickets a differential equation has been proposed. At this moment, we are at the testing phase of the agent; we are currently analyzing how the maximum income is guaranteed by depending on the constructed tour packages.

## References

1. Trading Agent Competiton, Game Description http://www.sics.se/tac/page.php?id=3
2. Sardinha J., Milidiú R., Paranhos P., Cunha P., Lucena C. *An Agent Based Architecture for Highly Competitive Electronic Markets*. Proceedings of the 18th Internanional FLAIRS Conference, pp. 326-331.
3. Toulis P., Kehagias D., Mitras P. *Mertacor: A Successful Autonomous Trading Agent*. AAMAS'06 (May 8-12, 2006, Hakodate, Hokkaido, Japan)
4. Wellman M., Reeves D., Lochner K., Suri R. *Searching for Walverine 2005*. IJCAI-05 Workshop on Trading Agent Design and Analysis, pp. 1-6, Nineteenth

International Joint Conference on Artificial Intelligence (30 July - 5 August, 2005, Edinburgh, Scotland).

5. He M., Jennings N., Prügel- Bennet A. *A Heuristic Bidding Strategy for Buying Multiple Goods in Multiple English Auctions.* ACM Transactions on Internet Technology, 2005.
6. Trading Agent Competition, TAC Agent Repository http://www.sics.se/tac/showagents.php