A Dynamic Pricing Approach in E-Commerce Based on Multiple Purchase Attributes

Tapu Kumar Ghose and Thomas T. Tran

School of Information Technology and Engineering
University of Ottawa, Ottawa, ON K1N 6N5, Canada
{tghos009,ttran}@site.uottawa.ca
http://www.site.uottawa.ca

Abstract. In this paper, we propose an approach of dynamic pricing where buyers purchase decision is dependent on multiple preferred purchase attributes such as product price, product quality, after sales service, delivery time, sellers' reputation. The approach requires the sellers, by considering the five attributes, to set an initial price of the product with the help of their prior knowledge about prices of the product offered by other competing sellers. Our approach adjusts the selling price of products automatically with the help of neural network in order to maximize seller revenue. The experimental results portray the effect of considering the five attributes in earning revenue by the sellers. Before concluding with directions for future works, we discuss the value of our approach in contrast with related work.

Keywords: Dynamic Pricing, Multiple Purchase Attributes, Electronic Commerce.

1 Introduction

In dynamic pricing products prices always respond to the fluctuation of the market and hence the prices keep on changing with the tick of a clock. Every seller wants to set the selling price of their products so that their revenue is maximized. Determining selling prices of products is a challenging task for the sellers to sustain in the market. The purpose of the dynamic pricing problem is to determine the selling prices such that sellers receive maximum revenue. Usually, a customer before buying a product selects a store/seller for the purchase. The selection may be done under multiple attributes (preferences), such as best price offered, after-sale services, product quality, delivery time, sellers' reputation etc. Therefore, the sellers have to provide a competitive price for a product in response to variation in the market parameters such as competitors' prices and consumers purchase preferences. There exist intelligent agents, called pricebots, which enable online sellers to dynamically calculate a competitive price for a product. According to Dasgupta et al. [8], "these intelligent agents provide a convenient mechanism for implementing automated dynamic pricing algorithms for the sellers in an online economy". However, some intelligent agents use a number of assumptions for the dynamic pricing in online markets. Some intelligent agents assume that sellers are provided with complete knowledge of market parameters, while some other agents consider product price as the only attribute that determines consumers' purchase decision [8]. In recent decades extensive research has been done in dynamic pricing. Some of the research made an assumption that there is only one seller in the market [16]. On the contrary, in real life sellers have limited or no prior knowledge about the market parameters (e.g., buyer's reservation price, competitive sellers' price and profit etc). In addition, in reality there exist several competitive sellers in online market.

The goal of this work is to address the problem of dynamic pricing in a competitive online economy, where a buyer's purchase decision is determined by multiple attributes. From the knowledge of our literature review [8,6,7], the most common attributes that can play vital role in determining customers' purchase decision would include product price, product quality, delivery time, after-sale service, and sellers' reputation. In our model we consider these mentioned five attributes in determining a competitive price for a product P. We use feed-forward neural network to determine a competitive price for the products in order to maximize sellers' revenue. In our simulation we showed that once the sellers set an initial price of the product, our model adjusts the price of the product automatically with the help of neural network in order to maximize profits. In setting the initial price of a product, we assume that sellers use their prior knowledge about the prices of the product offered by other competing sellers. The remaining of the paper is organized as follows: Section 2 provides background information on feed-forward neural network. Sections 3 discusses related work. Section 4 presents our proposed approach for dynamic pricing. Section 5 represents results and analysis from our simulation. Section 6 provides a brief discussion on our approach. Section 7 concludes the paper with future research directions.

2 Feed-Forward Neural Network

In feed-forward Neural Networks the nodes in input layer accept information from outside the network, while the nodes in output layer send information outside the network. Each node, also known as unit, is connected to one or more other nodes by directed links. Each link contains a numerical weight, for instance $W_{i,j}$ indicates the strength of the connection between unit i and unit j [9]. Each unit u_i has an activation value a_i which acts as output of the unit. The activation value is calculated as follows:

$$a_i = f\left(\sum_{j=0}^{i-1} W_{j,i} a_j\right) . \qquad (1)$$

where $\sum_{j=0}^{i-1} W_{j,i} a_j$ is the weighted sum of the inputs to unit u_i and f is the activation function applied to the weighted sum. We have chosen logistic sigmoid function as the activation function.

$$f(x) = \frac{1}{1 + \exp^{-x}}.$$

3 Related Works

Over the past few years there can be observed a noticeable rise in interest of dynamic pricing in commercial and research communities. In spite of rich literatures in the field, majority of the research works do not consider the competition markets [13]. Kephart et al. [1], for their work, considered a picture where a monopolist seller willing to maximize his/her revenue, provided buyers demand curve is random and unpredictable. Li et al. [17] studied the enterprises' dynamic decision problem on price strategies (dynamic pricing decision) in duopolistic retailing market under uncertain market state. Chinthalapati et al. [10] used machine learning based approach to study price dynamics in an electronic retail market. In the study they have taken price attributes into consideration that would determine a customer's buying decision. Dasgupta et al. [11] studied dynamic pricing in a multi-agent economy which consisted of buyers and competing sellers. They had taken price as the only attribute which take part in buyers purchase decision. In contrast, our model considers four more attributes (product quality, delivery time, after sale service and sellers' reputation) other than price. In fact our model is general enough to work for any number of attributes. Moreover, our model is not limited to two competitive sellers. Our model can work for both monopolist market and a competitive market with multiple sellers.

Dimicco et. al [5], by using Learning Curve Simulator, analyzed performance of two adaptive pricing algorithms: Goal-Directed (GD) and Derivative-Following (DF). They considered both monopoly and competitive economy of finite markets where goods like airlines ticket, sport events ticket, perishable goods have to be sold by finite time horizon. Alexandre et. al [14] discussed on the problems of dynamic pricing in finite time horizon. They considered a retailer who has to set the price of a good to optimize the total expected revenues over a period of time T. Their model is dependent on demand curve of the products. Kong [12], in his paper, examined seller strategies for dynamic pricing in a market for which a seller has finite time horizon to sell its inventory. Dasgupta et. al [15], in their paper, employed push strategies mechanism for dynamic pricing where they make use of demand curve. The authors considered time-limited goods in a supplier driven marketplace where goods are sold by maintaining strict deadline. On the contrary, our model of dynamic pricing does not require the sellers to figure out the demand curve of the products. Moreover, the model is not limited to goods with finite time horizon.

Greenwald and Kephart [2] explored no-regret learning for probabilistic pricing algorithm. In their model they considered an economy for single homogeneous goods. On the other hand, our model is not restricted to homogeneous goods. Our model is not concerned about how many sellers are there in the market, whereas, Tesauro and Kephart [3], in their experiment they assumed that there are only two competing sellers participating in the economy who alternatively take turns in adjusting their prices at each time step.

4 Design of the Proposed Model

In our model, for determining the price, we used a feed-forward neural network which contains three layers: input layer, hidden layer and output layer. The network we designed consists of five units in the input layer, one for each attribute mentioned in Section 1. The input layer also consists of one extra unit u_0 as the bias unit. We set the value of a_0 to the production cost of the product. Usually, sellers are not willing to sell their products below the production cost of the corresponding products. Hence, we considered the production cost of the product as the output of the bias unit. All the units accept numerical values as input. Initially, all the values a_1, a_2, a_3, a_4 and a_5 of the input units are set by the sellers. In setting the initial price of a product, we assume that sellers use their prior knowledge about the prices of the product offered by other competing sellers in the market.

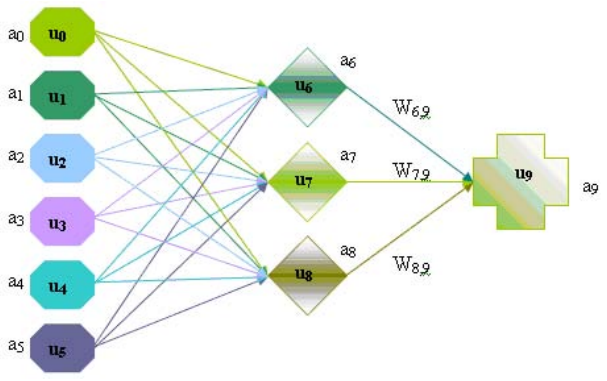

Fig. 1. A three-layered Feed-forward Neural Network for Price Determination

Our model can accept more attributes. One additional unit in the input layer needs to be added for each new attribute. On the other hand, in order to remove an attribute from the network the corresponding unit from the input layer, along with all the links that are connected to the unit, has to be eliminated. This implies that our model will work for any number of attributes.

4.1 Dynamic Pricing Algorithm

The price of the product determined by the network (Fig. 1) can be found by using final output a_9. The value of a_9 can be calculated with the aid of equation (1) as follows:

$$Final output, a_9 = f\left(W_{6,9}a_6 + W_{7,9}a_7 + W_{8,9}a_8\right). \tag{2}$$

$$\begin{aligned} where, a_6 &= f\left(W_{0,6}a_0 + W_{1,6}a_1 + W_{2,6}a_2 + W_{3,6}a_3 + W_{4,6}a_4 + W_{5,6}a_5\right) \\ a_7 &= f\left(W_{0,7}a_0 + W_{1,7}a_1 + W_{2,7}a_2 + W_{3,7}a_3 + W_{4,7}a_4 + W_{5,7}a_5\right) \\ a_8 &= f\left(W_{0,8}a_0 + W_{1,8}a_1 + W_{2,8}a_2 + W_{3,8}a_3 + W_{4,8}a_4 + W_{5,8}a_5\right) \end{aligned}$$

We sub-divide the process of dynamic pricing by our model of neural network into two phases: training phase and price determination phase. In the training phase we train our network with a set of training pattern. A training pattern consists of a set of inputs with desired output. A typical set of training pattern for our model would look as Table 1. Each row of Table 1 represents a training pattern which contains a set of inputs with corresponding desired output. Initially we assume that the buyers have equal preference on all the five attributes that we are considering. Therefore, we associate each link between input units and hidden units with equal weights. The purpose of the training process is to adjust the weights between the links such that the errors are minimized. To obtain this goal we feed units of the input layer of our network with the corresponding input values ($Input_{ij}$) from each training pattern. We then determine the output from our network and compare it with the corresponding desired output ($Output_i$) of the training pattern to calculate error. Finally, we update weights between the links depending on the calculated errors. In our model, during the process of training, the errors between the links are minimized by using back-propagation technique. The training process can be portrayed by the following steps:

i Input values from a training pattern to units of the input layer of the network.
ii If the current training pattern is the first training pattern of the training set, then associate the links between input units and hidden units with equal weight, i.e., 0.2. Also, associate the links between hidden units and output unit equally, i.e., 0.33.
iii Determine the value from the output layer.
iv Compute the error, i.e., the difference between desired output of the training pattern and the value obtained in step *iii*.
v If the error is more than zero then go to step *viii*.
vi If the error is approximately zero and there is more training pattern left, then take the next training pattern and go to step *i*.
vii If the error is approximately zero and there is no more training pattern left, then terminate the training process.

Table 1. General set of training pattern of our model

Product Price	Product Quality	Delivery Time	Sellers' Reputation	After Sales Service	Desired Output
$Input_{11}$	$Input_{12}$	$Input_{13}$	$Input_{14}$	$Input_{15}$	$Output_1$
$Input_{21}$	$Input_{22}$	$Input_{23}$	$Input_{24}$	$Input_{25}$	$Output_2$
$Input_{31}$	$Input_{32}$	$Input_{33}$	$Input_{34}$	$Input_{35}$	$Output_3$

viii Update the weights of the links using back-propagation technique to minimize the error.
ix Go to step *iii*.

The training phase updates weights between the links of the network as needed so that it can provide better output. Once the training process is complete, our model of network is ready to determine a competitive price for a product, P, from the price determination phase as follows:

i Set the production cost of the product, P as the input to bias unit of the input layer and set the weights of the links associated with bias unit to 1.
ii Set the values (a_i) of the input units for the corresponding purchase attributes of product (as mentioned in Section 3.1) by using prior knowledge about the prices of product offered by other competing sellers.
iii Run the network and derive the price from the output layer.
iv Set the price from the output layer as the product price.

4.2 Error Minimization and Price Determination

The error is minimized at each iteration from step *iii* through step *ix* of training phase. Once the price of a specific product is determined from the output layer from step *iv* of price determination phase, the weights of the links remain unchanged. In step *ii* of price determination phase we assume that sellers use their prior knowledge of price offered by other sellers in the market. This indicates that our model keep an eye on the competitive price set by other sellers in the competing market.

In dynamic online economy, the price of products keeps on changing with the tick of clock. In order to sustain in the competitive online economy a seller needs to update his/her price in response to price fluctuation by other competing sellers in the market. While updating the price by using our model, we go through training and price determination phase of our model to recalculate the price. Before recalculating the price we analyze the revenue earned by using the selling price, P_r, for the product, P, that was generated from our model. If the revenue earned is greater than zero, then in step *ii* of training phase instead of taking 0.2 as the weight, $W_{i,j}$, between the input units and the hidden units, we use the weights, $W_{i,j}$, that were determined during the last iteration of the training phase at the time of determining P_r and go through the process again. For instance, assume that the value of $W_{1,6}$ was 0.38 when the product price was determined from step *iv* of price determination phase. In such scenario, we would like to set the value of $W_{1,6}$ to 0.38 instead of 0.2 in step *ii* of training phase and run the process again. Moreover, at the time of determining P_r we store the values of input units from step *i* and values of output unit from step *iv* of price determination phase as the historical data. We use this historical data as an additional training set during the training phase. On the other hand, if there was no revenue earned then the entire process is run by providing a new set of inputs in step *ii* of price determination phase.

5 Results and Analysis

We simulated our model in an e-commerce market place to examine if the model performs better than the simple pricing algorithm outlined in the following subsection. We also analyzed if a seller earns more revenue by employing our model instead of the Derivative-Following (DF) strategy proposed in [10]. In derivative following (DF) strategy, initially, product prices are set randomly and profitability is observed. The product prices are increased in the same direction unless the observed profitability falls. If the observed profitability falls then product prices are decreased as long as profit is encountered. It requires keeping track of past average profit of each state, and increases the prices till the profitability level falls [10].

5.1 Simple Pricing Algorithm

A seller, by taking five attributes of our model into consideration, can employ a simple pricing algorithm to determine a competitive price of products. A simple pricing algorithm may take at least production cost of a product as initial selling price of the product. If a buyer prefers to enjoy any additional attributes such as after sale service of the product, then the algorithm may wish to add some additional price for each supplementary attributes. Finally, the algorithm would provide a selling price of the products. Since in online economy prices of products do not remain static, a seller has to update his/her offered price of the products. While updating the prices, there can arrive two different scenarios for a seller who employs the simple pricing algorithm. First, the algorithm, while updating the price by adding extra price for additional attributes, does not use any information of how other competing sellers in the market set their selling price. Since the algorithm has no knowledge about market parameters, it uses some random extra prices for additional attributes. Hence, the price can be too low or too high which may lead to earn inadequate revenue for a seller. In the second scenario let us assume that a seller who employed the simple algorithm, do some manual search on the prices offered by other vendors. The algorithm uses information obtained from the seller's search to determine a selling price for products. However, the manual search could be time consuming. It might take hours to days or even longer to gather information by manual search. Since prices change within very short span of time in online market, the information acquired by manual search during relatively large span of time might become outdated. Consequently, the algorithm would be using obsolete information which may lead to generate inappropriate output. In contrary, our model outputs a competitive selling price of products by providing importance to five attributes based on historical data, which implies it does not rely on any manual search. In addition, our technique, while determining competitive selling price, considers the sellers make use of their prior knowledge of the prices set by other competing sellers in providing initial input to the model. This indicates that our model keep an eye on the competitive price set by other sellers in competing market and utilizes fresh information of price.

5.2 Train Network

We assume that sellers use their historical data as the training patterns to the network. A training pattern consists of a set of inputs with desired output. We began our simulation by training the network of our model with 10 sets of training patterns so that errors can be minimized as much as possible by using back propagation algorithm. We trained our network for nine different numbers of epochs or iterations: 10, 50, 100, 500, 1000, 5000, 10000, 50000 and 100000. As the training continues, after each iteration or epoch, the network calculates amount of error. The calculated error is then used to update the weights of the links by using back propagation algorithm so that error is minimized in the next iteration. Practically the value of error never becomes zero, but approaches to zero. We let our network to tolerate an error of amount 0.01 and 0.001. We run our network with five different learning rates: 0.01, 0.005, 0.001, 0.0005 and 0.0001. Analysis of the training process in the following sub-section indicates that the model performs better if we use 50000 epochs with 0.005 learning rate during training the network.

5.3 Determine Training Parameters

The purpose of the model is to generate a competitive price for a product with respect to the price offered by other competing sellers in the market. The more number of training patterns are used to train the network in training phase, the better knowledge the model will contain. This would lead to generate a more competitive price. Hence, the performance of the model depends on the training phase. Besides number of training patterns used, the training process is dependent on three parameters: (i) number of epochs used, (ii) Learning rate and (iii) Error tolerance. Use of proper values for these parameters plays a vital role in the model's performance. We trained our model by using different values (as mentioned previous subsection) for these parameters with 10 sets of training patterns. Once training process is complete, our model is ready to use to determine a selling price for a product. We used our trained network to determine the price of a product (lets call it P). We then derive suitable values for the parameters by analyzing performance of the model through investigating the experimental results shown later in this section. Since our model requires initial selling price of the product to be set by the seller, we used the prices of Table 2 as initial selling price based on five different attributes of the product. We assumed the production cost of P is 645.00. According to our model, the output produced from the output layer of our model is considered as the selling price, P_r, at which a sellers, S, would be selling P. If there is M out of N buyers in the market willing to buy P at a cost of P_r, then the revenue earned by S can be calculated from the product of M and P_r.

While training our model, we found that the amount of revenue earned per product after selling it to a single buyer is closely identical to each other for different learning rates when lower number of epochs is used. Amount of revenue earned is increased gradually with higher number of epochs used. Another finding

Table 2. Initial Selling Price of Product P

Product Price	645.00
Product Quality	648.99
Delivery Time	745.99
After Sale Service	805.99
Sellers Reputation	718.99

was that that the model performs better if 0.01 is chosen as learning rate. The elapsed time for training our model increases gradually with increasing number of epochs. However, there is a rapid increase in elapsed time after 100,000 epochs. Therefore, we would not like to use more than 100,000 epochs during simulating an e-commerce market place in the following subsection. While training our model, we let our model to accept an error tolerance of 0.01 and 0.001. From 5000 number of epochs onwards the model delivers similar output for error tolerance of 0.01 and 0.001.

Under the above circumstances of analysis, for training our model during simulating an e-commerce market place in the following subsection we would like to use 100,000 epochs, 0.01 learning rate and 0.01 error tolerance.

5.4 Results

We simulated our model in an e-commerce market place to evaluate performance of our model. We consider a market place where three sellers (namely, $Seller_{simple}$, $Seller_{DF}$ and $Seller_{om}$) wish to sell a product P to 200 different buyers with five distinct preferable purchase attributes of products. $Seller_{simple}$ employs a **simple** pricing algorithm described in Section 5.1. $Seller_{DF}$ uses **d**erivative-**f**ollowing (**DF**) strategies and $Seller_{om}$ follows **o**ur **m**odel.

We run the market for ten rounds, with twenty buyers in each round. After each round we calculate the revenue earned by each seller. We then compare it with the revenue earned by the corresponding seller in previous round to determine the direction (positive or negative) of revenue earned. At the end of each round, we allow the sellers to update their selling prices. In DF strategies, the price of product is updated, by some amount, in the direction of revenue earned. We consider that $Seller_{DF}$ updates his/her selling price by a random amount between 0.00% and 5.00% of the current selling price. On the other hand, $Seller_{simple}$ updates the selling price either by using direction of revenue earned or by using information of prices set by other two sellers in the market during one of the previous rounds. We made an assumption that simple pricing algorithm performs manual search, which is time consuming, to gather information regarding other sellers' selling price. Therefore, the information may not be available to $Seller_{simple}$ at the end of each round. In addition, we assume that if the information is available, then due to manual slow searching process the information of the immediate previous round is not available to $Seller_{simple}$. For simplicity, we consider that if the information is available, then $Seller_{simple}$ updates the selling price of P by using average value of the prices set by other two sellers during the (r-2)th round, where r is the current round. In contrary, if the information is not available, $Seller_{simple}$ uses direction of revenue to update the price. We used a randomly-generated probability to determine if the information is available to $Seller_{simple}$. $Seller_{om}$ always uses our model to update the price. For simplicity we assume that in all ten rounds of the market there are equal numbers of buyers (four out of twenty) preferred each given five attributes.

We begin our simulation of the market by assuming that at the beginning of the first round $Seller_{simple}$ and $Seller_{DF}$ use data from Table 2 to set their selling price of the product P whose production cost is 645.00. $Seller_{om}$ uses information from Table 2 with 100,000 epochs, 0.01 learning rate and 0.01 error tolerance to generate a selling price (650.487) for P. Figure 2 summarizes the total revenue earned at the end of each round by the three different sellers who employed three distinct pricing algorithms.

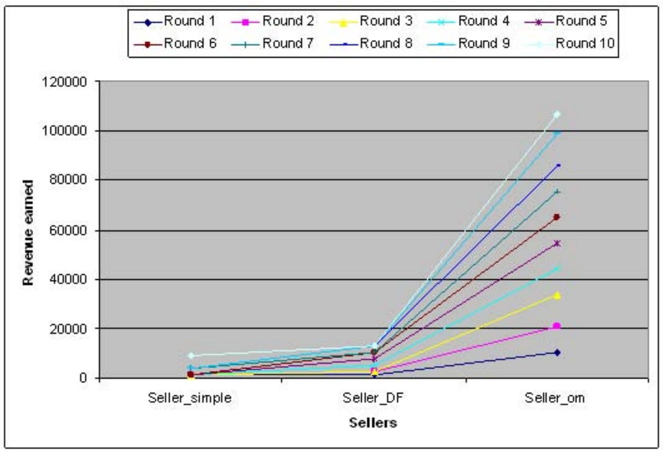

Fig. 2. Total revenue earned by three sellers

Initially, all the sellers managed to earn some revenue. Among the three sellers, the growth of revenue earned by $Seller_{simple}$ was the slowest. $Seller_{simple}$ failed to earn any revenue at the end of most of the rounds. Apart from first round, $Seller_{simple}$ earned some revenue after the end of seventh and tenth round. The performace of $Seller_{DF}$ in terms of earning revenue was better than $Seller_{simple}$, however, he/she could not beat $Seller_{om}$ in any of the rounds. On the contrary, $Seller_{om}$, who employed our model, earned revenue at each round. Moreover, after each round, $Seller_{om}$ earned higher revenue than that of revenue earned by other two sellers. At the end of tenth round, $Seller_{DF}$ earned nearly 43% more revenue than $Seller_{simple}$. On the other hand, $Seller_{om}$ earned nearly eight times more revenue than $Seller_{DF}$.

6 Discussion

The experimental results show that our model can attract more buyers compared to other two buyers, because we have considered multiple attributes in determining a selling price for the product P. Attracting more buyers from wider range of preferred attributes implies that more revenue can be earned. Various pricing algorithms are followed in present online economy. Among them game-theoric pricing (GT), myoptimal pricing (MY), derivative following (DF), and Q-learning (Q) are practiced widely. Game-theoretic (GT) strategy makes an assumption that all other competing sellers use game-theoretic [4]. However, in present world different sellers employ distinct pricing strategies. GT uses complete information regarding buyer population. Moreover, it does not use any historical data. In contrast, historical data plays an important role in understanding changing behavior of the market. We used little historical data of the price offered by other competitive sellers. In addition, our model is not concerned about what strategies are being used by other competing sellers. Similarly, My-optimal (MY) strategy does not dependant on whether other sellers employing different pricing strategies or not, but it is concerned about buyers demand curve and also the prices set by other sellers in the economy. MY also assumes that prices set by other competing sellers will remain unchanged [4]. On the contrary, sellers are always willing to change their offered price for the sake of sustaining in competing market. Hence, our model always keeps an eye on the random prices set by other sellers. Q-learning strategy is based on reinforcement learning and makes use of both buyers' demand curve and knowledge about competitors pricing strategies. On the other hand, our model does not rely on buyer demand curve. In short, we attempt to address the problem of dynamic pricing in a competitive online economy, where a buyer's purchase decision is determined by multiple attributes. By taking multiple purchase attributes into account we can attract more number of buyers which lead to earning more revenue.

7 Conclusion and Future Work

The proposed approach described here considered multiple purchase attributes to determing product price dynamically by using feed-forward neural network. We simulated an e-commerce market place with 200 buyers, three sellers where all the sellers trying to sell a product P. The experimental results showed that the seller employing our model earned higher revenue than that of earned by other two sellers who followed simple pricing algorithm and derivative-following (DF) strategies. We would like to compare our approach with other existing well known approach of dynamic pricing, like game-theoretic (GT), my-optimal (MY) etc. Our model made an assumption that sellers have limited prior knowledge about market parameters in setting the initial price of the products. We would like to eliminate the assumption from our model by employing a web crawler tool in our application in order to learn the information on prices set by other competitive sellers in the market. Using this information we may set the initial price of the products such that the sellers no need to initialize the product prices while using our model.

References

1. Kephart, J., Brooks, C., Das, R.: Pricing information bundles in a dynamic environment. In: ACM Conference on Electronic Commerce 2001, pp. 180–190 (2001)
2. Greenwald, A., Kephart, J.: Probabilistic pricebots. Agents, 560–567 (2001)
3. Tesauro, G., Kephart, J.: Foresight-based pricing algorithms in agent economies. Decision Support Systems 28(1-2), 49–60 (2000)
4. Greenwald, A., Kephart, J., Tesauro, G.: Strategic pricebot dynamics. In: ACM Conference on Electronic Commerce 1999, pp. 58–67 (1999)
5. DiMicco, J., Greenwald, A., Maes, P.: Dynamic pricing strategies under a finite time horizon. In: ACM Conference on Electronic Commerce, pp. 95–104 (2001)
6. Bar-Isaac, H., Tadelis, S.: Seller Reputation. Foundations and Trends in Microeconomics 4(4), 273–351 (2008)
7. Chen, Y., Tsao, C., Lin, C., Hsu, I.: A Conjoint Study of the Relationship between Website Attributes and Consumer Purchase Intentions. In: Pacific Asia Conference on Information Systems, PACIS (2008)
8. Dasgupta, P., Hashimoto, Y.: Multi-attribute dynamic pricing for online markets using intelligent agents. In: AAMAS (2004)
9. Russell, S.J., Norvig, P.: Artificial Intelligence: A modern Approach, 2nd edn. Prentice-Hall, Englewood Cliffs (2005)
10. Chinthalapati, V., Yadati, N., Karumanchi, R.: Learning Dynamic Prices in MultiSeller Electronic Retail Markets With Price Sensitive Customers, Stochastic Demands, and Inventory Replenishments. IEEE, Los Alamitos (2006)
11. Dasgupta, P., Das, R.: Dynamic Service Pricing for Brokers in a Multi-Agent Economy. IEEE, Los Alamitos (2000)
12. Kong, D.: One Dynamic Pricing Strategy in Agent Economy Using Neural Network Based on Online Learning. In: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (2004)
13. Luo, L., Xiao, B., Deng, J.: Dynamic pricing decision analysis for parallel flights in competitive markets, pp. 323–327. IEEE, Los Alamitos (2005)
14. Carvalho, A., Puterman, M.: Dynamic pricing and reinforcement learning. IEEE, Los Alamitos (2003)
15. Dasgupta, P., Moser, L., Melliar-Smith, P.: Dynamic Pricing for Time-Limited Goods in a Supplier-Driven Electronic Marketplace. Electronic commerce research (2005)
16. Gallego, G., Ryzin, G.: Optimal dynamic pricing of inventories with stochastic demand over finite horizons. Manage. Sci. 40(8), 999–1020 (1994)
17. Li, C., Wang, H., Zhang, Y.: Dynamic pricing decision in a duopolistic retailing market. In: Proceedings of the 6th World Congress on Intelligent Control and Automation, Dalian, China (June 2006)