Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

SEN

Software Engineering

Software ENgineering

An algorithm for on-line price discrimination

D.D.B. van Bragt, D.J.A. Somefun, E. Kutschinski,
J.A. La Poutré

REPORT SEN-R0213 JULY 31, 2002

CWI is the National Research Institute for Mathematics and Computer Science. It is sponsored by the Netherlands Organization for Scientific Research (NWO).
CWI is a founding member of ERCIM, the European Research Consortium for Informatics and Mathematics.

CWI's research has a theme-oriented structure and is grouped into four clusters. Listed below are the names of the clusters and in parentheses their acronyms.

Probability, Networks and Algorithms (PNA)

Software Engineering (SEN)

Modelling, Analysis and Simulation (MAS)

Information Systems (INS)

P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

ISSN 1386-369X

An Algorithm for On-Line Price Discrimination

D.D.B. van Bragt D.J.A. Somefun
E. Kutschinski J.A. La Poutré[†]
David.van.Bragt@cwi.nl Koye.Somefun@cwi.nl
Erich.Kutschinski@cwi.nl Han.La.Poutre@cwi.nl
[†] *Also with the School of Technology Management, Eindhoven University of Technology*
De Lismortel 2, 5600 MB Eindhoven, The Netherlands

CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

ABSTRACT
The combination of on-line dynamic pricing with price discrimination can be very beneficial for firms operating on the Internet. We therefore develop an on-line dynamic pricing algorithm that can adjust the price schedule for a good or service on behalf of a firm. This algorithm (a multi-variable derivative follower with adaptive step-sizes) is able to respond very quickly to changes in customers' demand. An additional advantage of the developed algorithm is that it does not require information about individual customers. Given the growing concern about customers' privacy this can be of great practical importance. Computational experiments (with different customer behavior models) indicate that our algorithm is able to successfully exploit the potential benefits of on-line price discrimination.

2000 Mathematics Subject Classification: 68T05, 90B99, 90C99
1998 ACM Computing Classification System: F.1.2, G.1.6, I.1.2, I.2.8, K.4.4
Keywords and Phrases: dynamic pricing, price discrimination, on-line learning, information goods, e-commerce, derivative follower algorithm
Note: Work carried out under theme SEN4 "Evolutionary Systems and Applied Algorithmics". This paper has been presented at the 8^{th} International Conference of the Society for Computational Economics on Computing in Economics and Finance (CEF'2002) (Aix-en-Provence, France, June 27-29, 2002). This research has been performed within the framework of the project "Autonomous Systems of Trade Agents in E-Commerce", which is funded by the Telematics Institute in the Netherlands.

1. Introduction

One aspect of the economics of the Internet is that the cost for changing prices is virtually zero. Consequently, it becomes economically feasible to change prices more frequently. This observation has resulted in a growing interest in on-line learning algorithms which, based on sales statistics, can dynamically and automatically adjust prices for offered goods [3, 4, 5, 1]. Another important characteristic of the economics of the internet is that selling information goods (e.g., computer software, electronic journals, stock quotes, music and video clips) requires high fixed and extremely low marginal costs. Due to this specific cost structure, firms operating in such a market are expected to enjoy some market power (cf. [7]) and, as a consequence, have some flexibility to set their own prices. A firm's major concern then remains the customers' willingness to pay for a product.

Differences in the willingness to pay (in the customer population) create an economic incentive for a firm to apply price discrimination, i.e., to charge different prices to different customers for the same goods and services. In general, however, a firm does not have complete information about the customers' willingness to pay.[1] This makes it difficult to successfully apply price discrimination. We therefore propose to combine on-line dynamic pricing with price discrimination. The advantage of

[1] In case of first-degree price discrimination, or perfect price discrimination, the firm has sufficient information to

using an on-line price discrimination algorithm is that, by automating the updating process, more frequent changes in the pricing scheme are possible. This can speed up the process of learning more profitable pricing schemes.

We introduce an efficient and robust algorithm for on-line price discrimination in this paper. As an example, this algorithm is used to dynamically adjust both the price for (bundles of) news items and the discounts for delayed delivery. The developed framework can also be applied to other types of price discrimination and/or other types of businesses such as, for example, the travel industry or the entertainment industry. Within the context of the travel industry an on-line learning algorithm can be used, for instance, to dynamically adjust both the price for an airfare, train ticket, or hotel room, and the size of the discounts for buying in advance. Within the entertainment industry, an on-line learning algorithm could be used, on the other hand, to develop, for instance, pricing schemes for video on demand (by offering flexible discounts for delayed viewing of a movie).

For the case considered in this paper (selling news items), it is reasonable to expect that (potential) customers have a higher valuation for a news item when it has been released more recently (compared to an "older" news item). The valuation of a customer for a bundle of news items can thus change significantly over time. Consequently, we need an on-line learning algorithm that is able to respond rapidly to changes in customers' valuations.

We therefore follow [3, 1, 6, 2, 9] by basing our algorithm for on-line price discrimination on the derivative follower (DF) algorithm. The DF algorithm is a (local) search algorithm that dynamically adjusts the price for the offered good based upon observed profits. This algorithm starts at some user-specified price level and then, step-by-step, changes the price in the same direction until the current profit drops relative to the profit in the previous trading period. In that case, the search direction is reversed and steps in the other direction are made. When the profit again decreases the search direction is reversed again, etc. This algorithm is able to react very quickly to changes in the profit landscape. Two additional advantages of the DF algorithm are that the underlying idea of the DF is very intuitive and that the DF requires very little problem specific knowledge (it is a "black-box" optimization technique). The latter makes it possible to develop an algorithm for on-line price discrimination which does not need information about individual customers. Given the growing concern about customers' privacy this can be of great practical importance.

A technical contribution of this paper is the development of an improved DF algorithm with an adaptive step-size (i-ADF). This algorithm has attractive (convergence) properties when operating in static and dynamic profit landscapes (unlike an ADF variant proposed by Dasgupta and Das [1]). The i-ADF algorithm is also extended in order to update multiple variables. This extension is important for dynamic pricing with price discrimination, where a list of prices should be adjusted (instead of a single price level). Computational experiments show that our multi-variable dynamic pricing algorithm is able to generate high profit levels for a variety of customer models.

The remainder of this paper is organized as follows. In Section 2, we discuss how the seller of a news item can use a dynamic pricing algorithm to update a price schedule. Moreover, we introduce our dynamic pricing algorithm and discuss the customer model that we use in the computational experiments. We report results from the experiments in Section 3. Section 4 concludes.

2. The Dynamic Pricing Model

2.1 The seller of a news item

In the market for news items there is one seller who offers a bundle of news items for sale during a number of consecutive trading periods. At the beginning of every trading period the seller announces a price schedule. Such a price schedule, denoted as $p_{t^*}(t)$, specifies the price if the news item is bought now (at time t^*) and delivered at time $t \geq t^*$. The seller can use the actual time of consumption to

determine the maximum willingness to pay for each customer and have them pay this amount. In case of second-degree price discrimination, or nonlinear pricing, the price depends on the amount purchased. In case of third-degree price discrimination, the firm can identify different customer groups who have a different willingness to pay [8].

apply price discrimination. For example, to have the good being delivered immediately after purchase can be more expensive than having the good being delivered later.

We model the price schedule as a step function with 3 steps (see Fig. 1).[2] Since a customer's

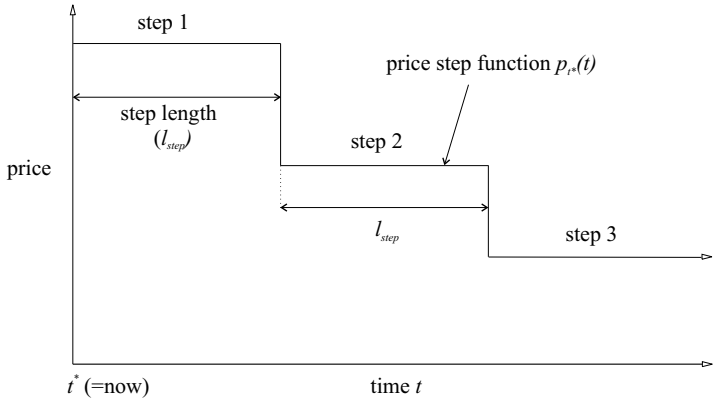

Figure 1: A price schedule with 3 steps.

valuation for a news item is (typically) negatively related to the delivery time we use price step functions with a negative slope. This requirement is enforced as follows. Assume we have 3 positive variables, labelled as x_1, x_2 and x_3, which are directly modified by the on-line dynamic pricing algorithm. We then calculate the price level for step no. i, p_i, as follows:

$$p_1 = x_1, p_2 = \frac{p_1}{1+\delta x_2}, p_3 = \frac{p_2}{1+\delta x_3} \quad (\delta > 0). \tag{2.1}$$

Since we assume that $x_i > 0$, this implies that $p_1 > p_2 > p_3$. We set δ equal to 0.04 in the experiments reported in Section 3.

2.2 The multi-variable DF algorithm

The seller can use a multi-variable DF algorithm to adjust this price step function over time. The four variables that are modified are x_1, x_2, x_3 and the length of a single step of the price function (l_{step}). Our multi-variable DF algorithm basically consists of four separate DF algorithms (one for each variable). We denote the DF algorithm which modifies variable x_i as DF_i (for $i = 1, 2, 3$) and the DF which modifies l_{step} as $\mathrm{DF}_{l_{step}}$.[3] These four DF algorithms do not work simultaneously, so only one variable is changed at each point in time.

The 4-variable DF algorithm starts by handing over the operation to DF_1. DF_1 will then change variable x_1 with a certain search step. This operation changes the offset of the price function (the prices of all steps are modified by changing x_1, see Eq. (2.1)). This process continues in the consecutive trading periods until the profit drops. The DF_1 algorithm then makes a final search step in the opposite direction. The 4-variable DF then passes the operation to DF_2 (which changes x_2). This operation adjusts the prices of step 2 and step 3. When the profit drops, a final search step is again made in the opposite direction. The 4-variable DF then passes the operation to DF_3 (leading to changes in the price of step 3). When DF_3 is also finished, l_{step} is modified by $\mathrm{DF}_{l_{step}}$. $\mathrm{DF}_{l_{step}}$ will now iteratively

[2] We also experimented with price schedules with more than 3 steps. Results were not better than those obtained with a function with 3 steps for the dynamic pricing problem considered in this paper. This is due to the time-consuming update process when a price schedule with many degrees of freedom is modified variable by variable (as we do in this paper). The usage of more complex price schedules may, however, be beneficial in special circumstances (e.g., when highly non-linear price schedules are very profitable, or when there is enough time to adjust a complex price schedule).

[3] All four variables should be positive numbers. Variables which become negative (or zero) are reset to unity by the DF algorithm. The DF continues to search in the positive direction with a (re-initialized) step-size of 0.5 in this case.

modify l_{step} in a certain direction as long as the profit does not drop (this changes the slope of the price step function). After $\mathrm{DF}_{l_{step}}$ is finished the whole process repeats itself. There is one exception to the rule for switching from one variable to the other: Whenever the profit immediately drops after a DF starts to change a particular variable, the DF returns to the initial search point and starts to search in the opposite direction (until it encounters a second drop in the profit).[4]

The above description of the multi-variable DF is, of course, incomplete without a description of the separate DF algorithms that modify the different variables. In this paper, these DF algorithms are instances of the improved adaptive step-size DF (i-ADF) algorithm. This algorithm is introduced in the next section.

2.3 The improved adaptive step-size DF (i-ADF)

The DF algorithm A DF algorithm iteratively modifies the price variable $x \in \mathbb{R}$ of a profit function $f(x,t)$. The variable $t \in \mathbb{Z}$ indicates that the profit function is typically changing over time. A DF algorithm increases or decreases the previous price $x(t-1)$ with a step-size $\sigma(t) > 0$ in the direction $\sigma_{dir}(t) \in \{-1,+1\}$. Suppose the DF increased the price at $t-1$ (i.e., $\sigma_{dir}(t-1) = +1$). The DF then continues to increase the price at time t if the profit at $t-1$ is not smaller than the profit at $t-2$. Otherwise the DF changes the search direction by setting $\sigma_{dir}(t)$ equal to –1. The DF then starts to decrease the price until $f(t-1)$ again becomes smaller than $f(t-2)$.

We say that a DF "turns" at time t whenever both $x(t+1)$ and $x(t-1)$ lie on the same side relative to $x(t)$ (i.e., either $x(t+1) > x(t)$ and $x(t-1) > x(t)$, or $x(t+1) < x(t)$ and $x(t-1) < x(t)$). We count the number of turns of the DF using the variable $k \in \mathbb{N}$ and say that the k^{th} turn occurs at $t = t(k)$.

Drawbacks of traditional DF algorithms There are a number of difficulties with current implementations of the DF algorithm. For example, the search step of the DF is typically fixed (i.e., $\sigma(t) = \sigma(t-1)$) [3, 6, 2, 9].[5] This can obviously result in a poor performance if the initial search point and/or the step-size are ill-chosen. A DF algorithm with a small step-size will, for instance, converge slowly to an optimum of the profit function if this optimum is located at a large distance (relative to the step-size of the DF) from the initial search point. Likewise, a large step-size may result in significant oscillations around the optimum.

This problem can be solved by using a DF algorithm with an adaptive step-size (ADF). Making the step-size adaptive can, however, potentially lead to unstable behavior of the algorithm. An example is the simple ADF (s-ADF) proposed by Dasgupta and Das [1]. This algorithm multiplies its step-size with a factor $\epsilon > 1$ if the profit increases (or remains the same). Otherwise the step-size is divided by ϵ (Appendix 1 contains the pseudo code which explains how the s-ADF adjusts its step-size over time). The difficulty with this (naive) approach is that the s-ADF does not take the distance to (local) optima of the profit function into account (relative to the current step-size). Hence, the s-ADF can expand its step-size in the vicinity of an optimum. This leads to convergence problems: the s-ADF algorithm is not able to converge to (local) optima of the profit function (see the analysis in Appendix 1).

An additional drawback of current DF and ADF implementations is that these algorithms have limited trend-following capabilities. That is, these algorithms can have difficulties to follow the peak of time-dependent profit functions. This ability to detect and follow trends is important in a dynamic pricing setup, where the profit function is typically changing rapidly over time.

Description of the i-ADF algorithm We developed an improved ADF algorithm (i-ADF) which solves the above-mentioned problems. The difference between the i-ADF and other implementations

[4] By default, each DF algorithm (re)starts by searching in the negative direction in the experiments reported in Section 3.

[5] In some of these studies the fixed step is multiplied by a random number (where the random number is drawn anew from a uniform distribution with range $[0,1]$ at each point in time). In the present study, this stochastic approach is not used (i.e., we focus on deterministic DF algorithms).

of the DF lies in how the search step-size is computed. Appendix 2 contains the pseudo code which explains how the i-ADF adjusts its step-size over time.

The i-ADF starts by increasing the step-size $\sigma(t)$ as long as it continues to search in the same direction. After a turn occurs, the i-ADF reduces its step-size. Because a turn typically signals that an optimum in the profit function has been passed, the i-ADF does not increase the step-size for a number of steps directly after a turn (to avoid a repeating "overshoot" of the optimum). An exception to this normal mode of operation occurs when the i-ADF turns twice in a row. In this case, the peak of the profit landscape may be moving over time. The i-ADF increases the step-size in this case to avoid the occurrence of premature convergence (due to a small step-size).

In the remainder of this section we will discuss some salient features of the i-ADF. On first reading, the more technical properties of the i-ADF can be skipped without consequence.

We start our discussion of the i-ADF by considering its behavior in case the profit function $f(x,t)$ is independent of time (i.e., when $f(x,t) = f(x)$) and unimodal in a sufficiently large interval.[6] In this case, we can say something about the distance between the optimum x_g and the price when a DF turns.

Lemma 1 *Suppose $f(x)$ is unimodal and let $t(k^*)$ denote the first point in time the optimum x_g has been passed by the DF. Then, whenever the DF turns at time $t \geq t(k^*)$, the distance $|x(t) - x_g| < 2\sigma(t)$.*

The intuition behind this result is that, after passing the (local) optimum x_g, the DF will at most make one more step in the same direction before it turns. Appendix 3 contains the proof of this lemma.

It is undesirable that the i-ADF increases its step-size in the vicinity of the optimum (this could lead to a failure to converge to the optimum). The i-ADF therefore keeps the step-size fixed for some steps after a turn. The i-ADF uses a total of 5 waiting periods. Lemma 2 shows that this is sufficiently long.

Lemma 2 *Suppose $f(x)$ is unimodal and let $t(k^*)$ denote the first time x_g has been passed by the i-ADF. Then, whenever the i-ADF turns at time $t \geq t(k^*)$, the i-ADF will not increment the step-size before it makes another turn.*

The intuition behind this result is as follows. After a turn, the i-ADF will always make 5 steps with a constant step-size. The i-ADF is constructed in such a way that the total length of the first 4 steps will always exceed the maximum distance to the optimum as defined in Lemma 1. That is, after at most 4 steps the optimum will be passed again. Thereafter, the i-ADF will at most make one more step in the same direction. As a consequence, the i-ADF will turn again after at most 5 steps without increasing the step-size. Appendix 3 contains the proof of this lemma.

Using Lemmas 1 and 2, we can proof (by induction on k) that the i-ADF algorithm is able to converge completely to (local) peaks of static profit functions.

Proposition 1 *Suppose the i-ADF passes a local optimum x_g of the profit function $f(x)$ and turns at time t. If $f(x)$ is unimodal in the interval $[x_g - 2\sigma(t), x_g + 2\sigma(t)]$, then the i-ADF converges to x_g.*

We note that it is not our primary objective to develop an algorithm that can be used to rapidly find the optimum of a unimodal function (other methods, like Fibonacci search, are better suited for this specific task). Nevertheless, guaranteed convergence for the class of unimodal functions is, in our opinion, desirable to qualify as a sound ADF algorithm.

The drawback of always decreasing the step-size after a turn is that the algorithm will, for instance, have difficulties to follow the peak of time-dependent profit functions. To circumvent this problem

[6] We say that a function $f(x)$ is unimodal in the interval $[a,b]$ if there exists a $x_g \in [a,b]$ such that is strictly increasing for all $x \in [a, x_g]$ and strictly decreasing for all $x \in [x_g, b]$. If a and b are not specified, we assume that $a = -\infty$ and $b = +\infty$.

we enriched the i-ADF with a simple heuristic that enables the i-ADF to detect rapid changes in the profit landscape. This heuristic uses the following observation.

Lemma 3 *Suppose the i-ADF makes a turn at time t (with $t \geq t(k^*)$). Then the i-ADF will not turn at time $t+1$ whenever the profit function $f(x)$ is unimodal.*

The intuition behind this lemma is that, after a turn, the i-ADF steps back into the direction of the point which was sampled at $t-1$. We also have that $f(t-1) \geq f(t)$ (otherwise the DF would not turn at time t). Lemma 3 then follows directly from the assumption of unimodality of $f(x)$. Appendix 3 contains the formal proof of this lemma.

Lemma 3 implies that the profit landscape is not unimodal when the i-ADF makes two successive turns. This leaves two possibilities: (i) the profit function is multimodal and/or (ii) the profit function is changing over time. Possibility (ii) is particularly important in our dynamic pricing setup, since the information good which is for sale tends to become less valuable as time passes by. In this case, the danger exists that the step-size of the i-ADF becomes too small to follow the peak of the moving profit function. The i-ADF will therefore *increase* the step-size whenever it records a sequence of dropping profits.

The multi-variable DF revisited We use, as we mentioned before, separate i-ADF algorithms to modify the different parameters of a price function. When optimizing such a multi-dimensional function, the distance to the optimum for a certain variable depends in general on the values of the other variables. This implies that a change in one of the variables (by a multi-variable DF) can strongly influence the distance to the optimum for the other variables.

Now suppose that a certain variable is modified by an i-ADF. Suppose also that this i-ADF is initially far removed from the optimum (measured with respect to its current step-size).[7] An advantage of the i-ADF is that, by increasing its step-size, the optimum can be approached rather quickly in this case. After it passes the optimum, the i-ADF will turn and make one step back in the direction of the optimum (after which another i-ADF starts to modify a different variable). If we now determine the distance to the optimum when the i-ADF started to operate, and the distance to the optimum when the i-ADF has finished, the latter distance is *always* smaller for a large class of functions. A precise statement of this key property of the i-ADF is given in Appendix 3 (see Proposition 3).

The multi-variable DF transfers the operation from one i-ADF to the other after an i-ADF turns. This rule for switching from one i-ADF to the other is used because it allows the separate i-ADFs to detect rapid changes in the profit landscape. If the profit drops immediately after the turn, it follows from Lemma 3 that the profit landscape is not unimodal. The i-ADF will then increase its step-size when it resumes operation. Consequently, the multi-variable i-ADF is also well equipped to follow the peak of time-dependent profit functions with multiple variables.

2.4 The customers

We now describe the basic customer behavior model that we use in the computational experiments. We distinguish between various customer classes. Customers that belong to the same class have an identical valuation for the news items. In every trading period, a customer from each class arrives at the (virtual) marketplace. If the customer does decide not to buy the information good then he will never reconsider buying the offered good (and effectively leave the marketplace).

Customers are assumed to have a reservation value for the offered news items that decreases linearly through time. They decide to buy the news items whenever their reservation value exceeds the price specified by the step function in one of the remaining trading periods. They consume the news items in the trading period where the difference between the reservation value and the price is maximized. In case of a draw they will pick the earliest trading period.

[7] Technically speaking, we mean with "far removed" that the optimum cannot be reached (or passed) in a single step.

More formally, we have the following quasilinear utility function:

$$U_i(q_i(t), t) = u_i(q_i(t)) - c_i \cdot q_i(t) \cdot t + m_i(t). \tag{2.2}$$

The term t denotes the delivery time of the news item. The quantity consumed by the i^{th} customer at time t is denoted as $q_i(t)$. Because we consider the brokerage of news items we have that $q_i(t) \in \{0, 1\}$ and $q_i(t) = 1$ for at most one point in time. Furthermore, $c_i > 0$ is a constant discount factor, and $m_i(t)$ denotes the amount of money held by the i^{th} customer. We set $u_i(0)$ equal to zero. We now assume that customers maximize their utility given the price step function $p_{t^*}(t)$. To buy a news item at time t means that the amount of money held by customer i is $p_{t^*}(t)$ less than when he does not buy the news item. Customer i will only buy the news item if there exist a $t \geq t^*$ such that $U_i(1, t) \geq U_i(0, t)$, otherwise not buying the item is the best option. (Recall that t^* denotes the current time.) If we define the valuation of customer i for the news item as $v_i(t) \equiv u_i(1) - c_i \cdot t$, this condition can be stated as $v_i(t) \geq p_{t^*}(t)$ (for a certain $t \geq t^*$). When the customer decides to buy the item, he chooses the earliest consumption period t which maximizes the term $v_i(t) - p_{t^*}(t)$.

Some assumptions made in the above customer model are relaxed in Section 3. We, for instance, also discuss results for a market with stochastic changes in the number of customers. We also treat the case where customers have nonlinear (instead of linear) utility functions.

3. Results

In a first series of computational experiments we distinguish between 30 different customer classes. The valuations of the customers in these classes are shown in Fig. 2. Note that three different customer

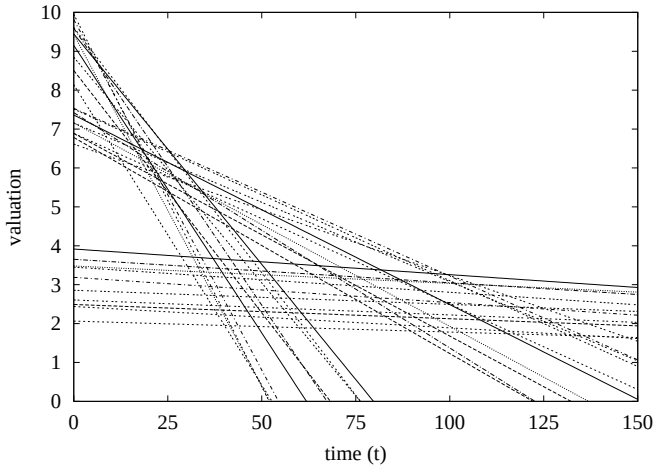

Figure 2: Valuations per customer class (for the experiments in Fig. 3).

types can clearly be distinguished among the 30 customers shown in Fig. 2: (i) a group of "impatient" customers, who are willing to pay much for early delivery but much less later on, (ii) a group of "patient" customers who value the news item rather high initially, with only a slow decay over time, and (iii) a group of "indifferent" customers who are not willing to pay much, more or less independent of the delivery time.

Figure 3 shows an example of the adjustment of a price step function with 3 steps by the 4-variable i-ADF algorithm (for the customer valuations in Fig. 2). Figure 3a shows the adaptation of the different steps of the price function (see Fig. 1). Figure 3b shows the adaptation of the price function's step length (i.e., l_{step}). The actual price schedules after 20, 60, and 100 trading periods are shown in Fig. 3c.

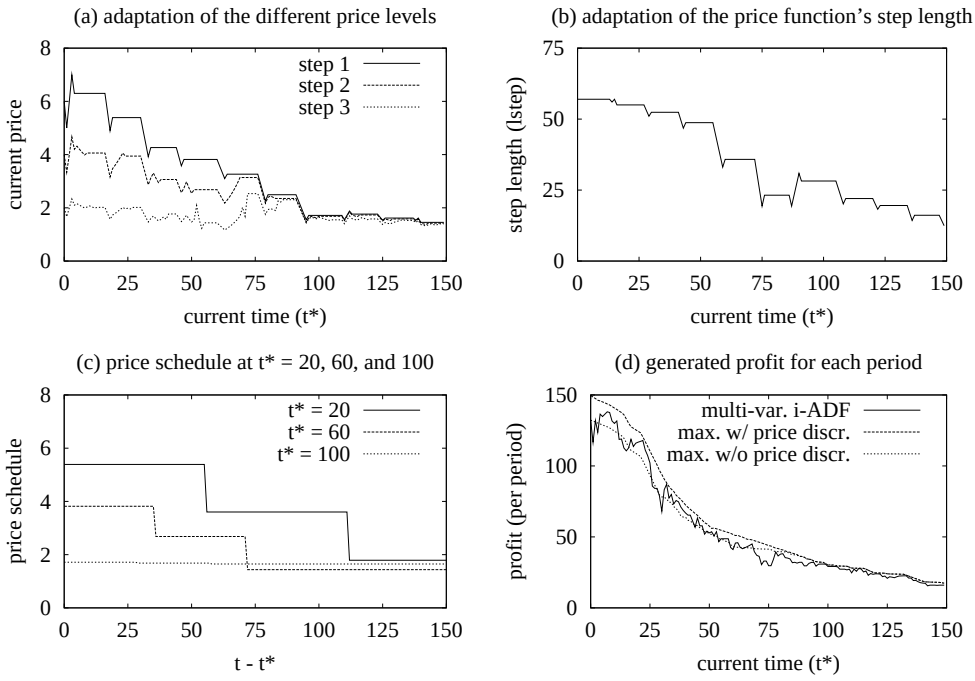

Figure 3: On-line price discrimination with the multi-variable i-ADF.

In Fig. 3d, we compare the profits generated in this experiment with the maximum profit that can be reached theoretically with a price function with 3 steps. We also show the maximum profits that can be reached theoretically when price discrimination is not applied (i.e., when the seller posts a single price at each point in time). Note that price discrimination is potentially beneficial in the first 90 trading periods. Thereafter, the theoretical optima with and without price discrimination coincide.

At the beginning of the experiment, the multi-variable i-ADF needs several "learning periods" to adjust and improve the initial price schedule. The performance of the algorithm increases rapidly, however. Figure 3d shows for instance that highly profitable pricing schemes are already generated by the algorithm after only 8 trading periods. At later points in time (especially around $t^* = 22, 32, 45$ and 68) performance of the multi-variable i-ADF remains high (note that at these points in time the generated profits are significantly higher than the maximum profits that can be reached without applying price discrimination).

In the long run, applying price discrimination is no longer advantageous for the seller. Interestingly, the multi-variable i-ADF generates almost entirely flat price schedules in this case (see the price function for $t^* = 100$ in Fig. 3c). The algorithm is thus able to detect automatically that simple flat price schedules perform optimally in the long run.

Additional computational experiments demonstrate that the multi-variable i-ADF performs significantly better than an algorithm which consists of DFs with a fixed step-size (especially when the initial prices or the initial step-sizes are ill-chosen). We also investigated whether the multi-variable i-ADF continues to perform well for alternative customer models. The following conclusions can be drawn from this study:

- *Customers with nonlinear valuations.* The multi-variable i-ADF performs well in case of nonlinear customer valuations (e.g., valuations with an exponential decay over time, or valuations which decrease as a quadratic function). This (positive) result is not surprising since (i) this algorithm makes no assumptions about the customers' behavior (it is a "black-box" optimizer)

and (ii) the i-ADF algorithm is able to respond quickly to changing customer demand.

- *A large variety of customer types.* The performance of the multi-variable i-ADF remains high when the number of different customer types increases. Because we use price functions with a negative slope, the risk of overestimating the mean customer valuation is small since customers with a low valuation can buy the item at a later point in time (when the price is low).
- *Stochastic entry of new customers.* The performance of the multi-variable i-ADF remains high when the number of new customers randomly fluctuates at each trading period. We used the mean profit (averaged over all new customers) as the input for the algorithm in this case.

Summarizing, we can conclude that the multi-variable i-ADF is a suitable algorithm for on-line price discrimination. Its strengths are in particular: (i) the ability to adapt the search step on-line, (ii) the ability to operate successfully in case of rapidly changing profit landscapes, and (iii) the potential to adjust multiple parameters in case of price functions with multiple degrees of freedom.

4. Conclusions

The combination of on-line dynamic pricing with price discrimination can be very beneficial for firms operating on the Internet (e.g., firms selling information goods). We have therefore investigated if it is possible to develop a dynamic pricing algorithm that can successfully adjust the price schedule for a good or service on behalf of a firm. Such a pricing algorithm should meet several criteria in order to generate a high total profit under a variety of (possibly changing) market circumstances. For example, it is desirable that the algorithm makes frequent and automated updates of the pricing scheme, and is able to cope with a variety of customers with changing preferences.

In this paper, we have developed a dynamic pricing algorithm which meets these criteria. This algorithm (a multi-variable derivative follower with adaptive step-sizes) generates highly profitable pricing schemes even if customers' demand is changing over time. An additional advantage of the developed algorithm is that it does not require information about individual customers. Given the growing concern about customers' privacy this can be of great practical importance.

As an example, the developed pricing algorithm has been used to dynamically adjust both the price for (bundles of) news items and the discounts for delayed delivery. A series of computational experiments show that the algorithm is able to generate high profit levels for a variety of customer behavior models. The developed framework can also be applied to other types of price discrimination and/or other types of businesses such as, for example, the travel or entertainment industry.

Acknowledgements

This research has been performed within the framework of the project "Autonomous Systems of Trade Agents in E-Commerce", which is funded by the Telematics Institute in the Netherlands.

APPENDICES

1. Description and analysis of the s-ADF

Algorithm 1 shows the adaptation of the step-size $\sigma(t)$ as proposed in [1].

Algorithm 1 Adaptation of the step-size $\sigma(t)$ by the s-ADF algorithm [1]. We assume that $\epsilon > 1$.

```
1 if (f(t-1) ≥ f(t-2)) //profit does not decrease
2   σ(t) = ε * σ(t-1) //increase the step size
3 else //profit decreases
4   σ(t) = σ(t-1)/ε //decrease the step size
```

Dasgupta and Das claim that this simple step-size adaptation mechanism is able to "reduce the step size when the price is in the vicinity of the optimum" [1, p. 302]. We show in this appendix, however, that such a general claim cannot be made. In fact, the price can fail to converge to the optimal level when Algorithm 1 is used to adjust the search step. An example is given in Proposition 2 for the case of simple (unimodal and symmetric) profit functions. We assume in Proposition 2 that Algorithm 1 is used in combination with a deterministic DF (Dasgupta and Das use a stochastic DF).

Proposition 2 *Assume the profit function $f(x,t)$ is independent of time, unimodal, and symmetric with respect to x_g (i.e., $f(x_g - \Delta x) = f(x_g + \Delta x)$). Assume also that at time $t = 0$ the s-ADF arrives at $x(0) = x_g - \theta\sigma(1)$. Let $\sigma_{dir}(1) = +1$ and let $\theta \in [\frac{1}{2}, 1)$. Also assume that $\theta \leq \frac{\epsilon}{2}$. The maximum distance to x_g then* **increases** *over time. More precisely, for each point in time t there exists a $t' \leq t$ such that the distance $|x(t') - x_g| \geq \frac{1}{2}\sigma(1)\epsilon^{2\lfloor t/6 \rfloor}$.*

Appendix 3 contains the proof of this proposition. Because $\theta \in [\frac{1}{2}, 1)$, we have that $x(0) < x_g$, $x(1) > x_g$, and $x(2) > x(1)$. The s-ADF will thus accelerate once after passing the optimum. The condition $\theta \leq \frac{\epsilon}{2}$ implies that the ADF accelerates when it returns to $x(0)$ at $t = 4$. If this condition is not satisfied, the ADF will turn at $t = 4$ and continue to sample the same sequence of values. If $\theta \in [0, \frac{1}{2})$, the s-ADF turns directly after passing the optimum. It is easy to see that the s-ADF then also returns to the initial point (i.e., x(0)) after 4 iterations and continues to sample the same sequence of x values. Because we assume that $f(x)$ is symmetric, a very similar proposition can be stated for the case where $x(0) > x_g$, $x(1) < x_g$, and $x(2) < x(1)$.

This analysis thus shows that the s-ADF algorithm can fail to converge to the optimal price level, even in case of simple (unimodal) search problems.

2. Technical description of the i-ADF

Algorithm 2 shows how the step-size $\sigma(t)$ is adapted by the i-ADF algorithm. The variable t_{init}

Algorithm 2 Adaptation of the step-size $\sigma(t)$ by the i-ADF algorithm. We assume that $1 < \alpha \leq 2$ and $\beta, \gamma > 1$.

```
1 if (f(t-1) ≥ f(t-2)) //profit does not decrease
2   if (k = 0) σ(t) = β * σ(t-1) //initial phase
3   else if (k ≥ 1 ∧ t - t(k) > 5) σ(t) = β * σ(t-1)
      //only increase the step-size after a turn
      //if t > t(k) + 5
4   else σ(t) = σ(t-1) //keep the step-size fixed
5 else //profit decreases
6   if (t = t_init + 2) σ(t) = σ(t-1) //return to start
7   else if (k ≥ 2 ∧ t(k) = t(k-1) + 1) σ(t) = γ * σ(t-1)
      //increase step-size after two successive turns
8   else σ(t) = σ(t-1)/α //decrease the step-size
```

denotes the point in time when the i-ADF starts to operate. This variable is reset (to the current time) each time the operation is transferred to the i-ADF by a multi-variable i-ADF algorithm.

We use the following settings for the i-ADF algorithm in this paper: $\alpha = 1\frac{3}{7}$, $\beta = 1.6$, and $\gamma = 2.0$. These settings satisfy the two conditions stated in Proposition 3. We also verified in a parametric study that these values are chosen properly for the dynamic pricing problem at hand.

3. Proofs

Proof of Proposition 2. We assume that the profit function $f(x,t)$ is independent of time, unimodal, and symmetric. The proof uses the following lemma.

Lemma 4 *Let* $x(t) = x_g - \theta(t)\sigma(t+1)$. *Let* $\sigma_{dir}(t+1) = +1$ *and let* $\theta(t) \in [\frac{1}{2}, 1)$. *Also assume that* $\theta(t) \leq \frac{\epsilon}{2}$. *We then have that* $x(t+6) = x_g - \theta(t+6)\sigma(t+7)$, *with* $\theta(t+6) \in [\frac{1}{2}, 1)$, $\theta(t+6) \leq \frac{\epsilon}{2}$, $\sigma(t+7) = \epsilon^2\sigma(t+1)$ *and* $\sigma_{dir}(t+7) = \sigma_{dir}(t+1)$.

PROOF. The s-ADF will pass the optimum, accelerate at $t+1$, turn at $t+2$, accelerate at $t+3$ and $t+4$ (because $\theta \leq \frac{\epsilon}{2}$ we have that $|x(t+4) - x_g| \leq |x(t+3) - x_g|$), turn at $t+5$, and accelerate at $t+6$. Because the s-ADF turns twice between t and $t+6$, $\sigma_{dir}(t+7) = \sigma_{dir}(t+1)$. Also, because the s-ADF accelerates at $t+1, t+3, t+4$, and $t+6$, we have that $\sigma(t+7) = \epsilon^2\sigma(t+1)$.

We now prove that $\theta(t+6) \in [\frac{1}{2}, 1]$. By inspecting the iterations made by the s-ADF, we find that $x(t+6) = x_g - [\theta(t) + \epsilon(\epsilon-1)]\sigma(t+1)$ and $x(t+7) = x_g + [\epsilon - \theta(t)]\sigma(t+1)$. Because $\epsilon > 1$ and $\theta(t) \in [\frac{1}{2}, 1)$, $x(t+6) < x_g$ and $x(t+7) > x_g$. This implies that $\theta(t+6) \in (0, 1)$. Now assume that $\theta(t+6) \in (0, \frac{1}{2})$. In this case, $|x(t+6) - x_g| < |x(t+7) - x_g|$. Substituting (and re-arranging terms) then yields the following inequality: $\epsilon^2 - 2\epsilon + 2\theta(t) < 0$. This inequality is violated if $\theta(t) \geq \frac{1}{2}$. We assume, however, that $\theta(t) \in [\frac{1}{2}, 1)$. By contradiction, we have thus shown that $\theta(t+6) \in [\frac{1}{2}, 1)$.

Finally, we need to proof that $\theta(t+6) \leq \frac{\epsilon}{2}$ if $\theta(t) \leq \frac{\epsilon}{2}$. Assume that $\theta(t+6) > \frac{\epsilon}{2}$. Since $\theta(t+6) = \frac{[\theta(t)+\epsilon(\epsilon-1)]}{\epsilon^2}$ is a strictly increasing function of $\theta(t)$, we can restrict our attention to the case where $\theta(t) = \frac{\epsilon}{2}$. Substituting and re-arranging then yields the following inequality: $\epsilon^2 - 2\epsilon + 1 < 0$. This inequality is false, so we have derived a contradiction.

QED

Assume that the conditions stated in Lemma 4 are fulfilled at $t = 0$. We now define a "cycle" as a sequence of 6 steps by the s-ADF. We count the number of completed cycles at time t by the index $m(t) \in \mathbb{N}$. So, by definition, $m(t) \equiv \lfloor t/6 \rfloor$. Now denote the last point in time when a new cycle started as $t^{'}(t) \equiv 6m(t)$. Note that $t^{'}(t) \leq t$. By induction, it follows from Lemma 4 that $\sigma(t^{'}(t)+1) = \epsilon^{2m(t)}\sigma(1)$. Moreover, because $\theta(t^{'}(t)) \in [\frac{1}{2}, 1)$, the distance $|x(t^{'}(t)) - x_g| \geq \frac{1}{2}\sigma(t^{'}(t)+1)$. For every $t \geq 0$, there thus exists a $t^{'} \leq t$ such that $|x(t^{'}) - x_g| \geq \frac{1}{2}\sigma(1)\epsilon^{2\lfloor t/6 \rfloor}$.

Proof of Lemma 1. We have to distinguish between two cases:

(a) Either $x(t-1) = x_g$, or $x(t-1)$ lies on the other side as $x(t)$ relative to x_g (i.e, $x(t) > x_g$ and $x(t-1) \leq x_g$, or $x(t) < x_g$ and $x(t-1) \geq x_g$). In this case we have that $|x(t) - x_g| \leq \sigma(t)$.

(b) $x(t-1)$ lies on the same side as $x(t)$ and $x(t-2)$ lies on the other side as $x(t)$ (relative to x_g). In this case we have that $|x(t) - x_g| < \sigma(t-1) + \sigma(t) \leq 2\sigma(t)$.

To see that these are the only two cases, suppose that neither case (a) nor case (b) holds. This means that $x(t-1)$ lies on the same side as $x(t)$ (relative to x_g) and either $x(t-2) = x_g$ or $x(t-2)$ also lies on the same side as $x(t)$ (i.e., either $x(t-2) \geq x_g$, $x(t-1) > x_g$, and $x(t) > x_g$; or $x(t-2) \leq x_g$, $x(t-1) < x_g$, and $x(t) < x_g$). Given the assumption that the i-ADF turns at time t we will obtain a contradiction. Observe that because $x(t-1)$ and $x(t)$ lie on the same side (relative to x_g) the inequality

$$|x(t-1) - x_g| < |x(t) - x_g| \tag{3.1}$$

holds (otherwise the i-ADF does not turn at time t). Then two cases remain:

(i) $|x(t-2) - x_g| < |x(t-1) - x_g|$, so it follows from the assumption that f is unimodal that the i-ADF turns at $t-1$. This implies that $|x(t-1) - x_g| > |x(t) - x_g|$ which contradicts Eq. (3.1).

(ii) $|x(t-2) - x_g| > |x(t-1) - x_g|$, so the i-ADF does not turn at $t-1$. Therefore $|x(t-1) - x_g| > |x(t) - x_g|$, which contradicts Eq. (3.1).

Proof of Lemma 2. Suppose the i-ADF makes the k^{th} turn at time t. This means that the step size is divided by a factor α at $t+1$, i.e., $\sigma(t+1) = \frac{1}{\alpha}\sigma(t)$ (see Algorithm 2, line 8). The i-ADF will then multiply the step size with a factor β at $t+6$ if the search direction does not change (see Algorithm 2, line 3). We will show below that if $\alpha \leq 2$ (as we assume) then the i-ADF always turns before $t+6$. Consequently, the step size is not incremented and we have that $\sigma(t(k+1)) = (\frac{1}{\alpha})\sigma(t(k))$.

Suppose the i-ADF does not change the search direction before $t+6$. This means that $|x(t+4) - x(t)| = \frac{4}{\alpha}\sigma(t) \geq 2\sigma(t)$. Since, $|x(t) - x_g| < 2\sigma(t)$ (see Lemma 1), we can conclude that $x(t+4)$ lies on the other side of x_g as $x(t)$. Moreover, $|x(t+5) - x(t)| = \frac{5}{\alpha}\sigma(t)$ and $x(t+5)$ lies on the same side as $x(t+4)$ relative to x_g. Since $|x(t+5) - x_g| > |x(t+4) - x_g|$, it then follows from the assumption that $f(x)$ is unimodal that $f(x(t+5)) < f(x(t+4))$. Consequently, the i-ADF will turn at $t+5$.

Proof of Lemma 3. Observe that either $x(t+1)$ lies between x_g and $x(t)$, or $x(t+1)$ lies between x_g and $x(t-1)$. In the first case, it follows from the unimodality of $f(x)$ that $f(x(t+1)) > f(x(t))$, hence the i-ADF will not turn at time $t+1$. In the second case, it follows from the unimodality of $f(x)$ that $f(x(t+1)) > f(x(t-1))$. Moreover, since the i-ADF turns at time t we have that $f(x(t-1)) > f(x(t))$. Thus, $f(x(t+1)) > f(x(t))$ and the i-ADF will not turn at time $t+1$.

Proof of Proposition 3. Throughout the remainder of this appendix we assume that the n-variable i-ADF operates on an n-unimodal function $f : \mathbb{R}^n \to \mathbb{R}$.

Definition 1 (n-unimodal) *Let $f : \mathbb{R}^n \to \mathbb{R}$, with $n > 1$. Moreover, let $f_{x_{-i}} : \mathbb{R} \to \mathbb{R}$ denote the function that is obtained by fixing all but the i^{th} variable in f, where the $n-1$ fixed variables are equal to $x_{-i} \equiv (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$. We now say that f is n-unimodal if f has a unique global maximum and if, for all $i \in \{1, \ldots, n\}$, $f_{x_{-i}}$ is a unimodal function for any $x_{-i} \in \mathbb{R}^{n-1}$.*

Intuitively, a function of n variables is n-unimodal if it has a unique global maximum and if by varying only one of its variables the resulting unary function is unimodal (i.e., this unary function has a unique global maximum).

For the proof of Proposition 3 some additional notation is useful. We denote the i-ADF algorithm that modifies variable i as i-ADF$_i$. We let k denote the number of times the n-variable i-ADF has switched from modifying one variable to another. Moreover, $t(k)$ denotes the point in time when the n-variable i-ADF turns and switches for the k^{th} time to another variable (one step later). If $k \in K_i$, with $K_i = \{(i-1), (n+i-1), (2n+i-1), (3n+i-1), \ldots\}/\{0\}$, then $t(k)+1$ denotes the point in time when the i-ADF$_i$ algorithm starts to modify variable i. It is convenient to have $t_k \equiv t(k)$ and $t_{k+1} \equiv t(k+1)$. Additionally, $x_{i,g}(t)$ denotes the currently (given x_{-i}) optimal value of x_i. Furthermore, $\Delta x_{i,g}(k) = |x_{i,g}(t_k+1) - x_i(t_k+1)|$, i.e., $\Delta x_{i,g}(k)$ denotes the distance of x_i to the optimal value at t_k+1. Finally, let $\sigma_i(t)$ denote the step size of the i-ADF$_i$.

Note that an i-ADF$_i$ does not pass on the operation of the i-ADF whenever it makes a turn immediately after it starts operating. In this case k is not updated and the current time t is set equal to $t(k)+1$ again.

Proposition 3 *The n-variable i-ADF operates on an n-unimodal function. Additionally, suppose that (1) $\alpha_i > (\alpha_i - 1)\beta_i$ and (2) $2\alpha_i > \beta_i^2(\alpha_i - 1) + \beta_i$ hold for $i = 1, \ldots, n$. If $k \in K_i$ and $\sigma_i(t_k+2) \leq \frac{1}{2}\Delta x_{i,g}(k)$ we then have that $\Delta x_{i,g}(k+1) < \Delta x_{i,g}(k)$.*

PROOF. We distinguish between two cases:

(a) either $x_i(t_{k+1}+1)$ lies on the opposite side as $x_i(t_{k+1})$, relative to $x_{i,g}(t_k+1)$, or $x_i(t_{k+1}+1) = x_{i,g}(t_k+1)$. By construction of the i-ADF$_i$, $x_i(t_{k+1}+1)$ lies between $x_i(t_{k+1}-1)$ and $x_i(t_{k+1})$. Consequently, both $x_i(t_{k+1}-1)$ and $x_i(t_{k+1}+1)$ do not lie on the same side as $x_i(t_{k+1})$, relative to $x_{i,g}(t_k+1)$. Furthermore, $x_i(t_{k+1}+1)$ lies closer to $x_{i,g}(t_k+1)$ than $x_i(t_{k+1}-1)$. Thus $\Delta x_{i,g}(k+1) < \Delta x_{i,g}(k)$.

(b) $x_i(t_{k+1}+1)$ lies on the same side as $x_i(t_{k+1})$, relative to $x_{i,g}(t_k+1)$. We distinguish between two subcases.

(b1) Assume that either $x_i(t_{k+1}-1)$ lies on the opposite side as $x_i(t_{k+1}+1)$ relative to $x_{i,g}(t_k+1)$, or $x_i(t_{k+1}-1) = x_{i,g}(t_k+1)$. Since $\sigma_i(t_k+2) \leq \frac{1}{2}\Delta x_{i,g}(k)$ it follows from (b1) that $\sigma_i(t_{k+1}-1) <$

$\Delta x_{i,g}(k)$. Moreover, since $x_i(t_{k+1}+1)$ lies between $x_i(t_{k+1})$ and $x_i(t_{k+1}-1)$ we have

$$\begin{aligned}\Delta x_{i,g}(k+1) &\leq |x_i(t_{k+1}+1) - x_i(t_{k+1}-1)| \\ &\leq \beta_i\sigma_i(t_{k+1}-1) - \frac{\beta_i}{\alpha_i}\sigma_i(t_{k+1}-1) \\ &= \frac{\beta_i}{\alpha_i}(\alpha_i - 1)\sigma_i(t_{k+1}-1).\end{aligned}$$

Hence it suffices if $\frac{\beta_i}{\alpha_i}(\alpha_i - 1) < 1$, i.e., $\alpha_i > (\alpha_i - 1)\beta_i$, which follows from condition (1).

(b2) Assume that $x_i(t_{k+1}-1)$ lies on the same side as $x_i(t_{k+1}+1)$, relative to $x_{i,g}(t_k+1)$. Let $m \equiv t_{k+1} - (t_k+1)$. From the assumption that $\sigma_i(t_k+2) \leq \frac{1}{2}\Delta x_{i,g}(t_k+1)$, and the fact that $x_i(t_{k+1}-1)$ lies on the same side as $x_i(t_{k+1}+1)$, it follows that $m > 3$. For $m = 4$ we have that

$$\begin{aligned}\Delta x_{i,g}(k) &> \sigma_i(t_k+2) + \sigma_i(t_k+3) \\ &= 2\sigma_i(t_k+2)\end{aligned}$$

and

$$\begin{aligned}\Delta x_{i,g}(k+1) &< \sigma_i(t_k+4) + \sigma_i(t_k+5) - \sigma_i(t_k+6) \\ &= \left(2 - \frac{1}{\alpha_i}\right)\sigma_i(t_k+2).\end{aligned}$$

Hence, $\Delta x_{i,g}(k) > \Delta x_{i,g}(k+1)$. Analogously, for $m = 5$ we have that

$$\begin{aligned}\Delta x_{i,g}(k) &> \sigma_i(t_k+2) + \sigma_i(t_k+3) + \sigma_i(t_k+4) \\ &= 3\sigma_i(t_k+2)\end{aligned}$$

and

$$\begin{aligned}\Delta x_{i,g}(k+1) &< \sigma_i(t_k+5) + \sigma_i(t_k+6) - \sigma_i(t_k+7) \\ &= \left(1 + \frac{\beta_i}{\alpha_i}(\alpha_i - 1)\right)\sigma_i(t_k+2).\end{aligned}$$

Thus for $\Delta x_{i,g}(k) > \Delta x_{i,g}(k+1)$ to be the case it suffices if $2\alpha_i > \beta_i(\alpha_i - 1)$, which is the case since $\alpha_i > (\alpha_i - 1)\beta_i$ (condition (1)). Finally, for $m \geq 6$ we have that

$$\Delta x_{i,g}(k) > (4 + (\sum_{n\geq 6}^{m} \beta_i^{n-6} - 1))\sigma_i(t_k+2)$$

and

$$\Delta x_{i,g}(k+1) < \left(1 + \frac{\beta_i}{\alpha_i}(\alpha_i - 1)\right)\beta_i^{m-5}\sigma_i(t_k+2).$$

We can show by induction on m (for $m \geq 6$) that if $4\alpha_i > \beta_i^2(\alpha_i - 1) + \beta_i\alpha_i$ and $2\alpha_i > \beta_i^2(\alpha_i - 1) + \beta_i$ (condition (2)) hold then $\Delta x_{i,g}(k) > \Delta x_{i,g}(k+1)$. Moreover, combining $\alpha_i > (\alpha_i - 1)\beta_i$ (condition (1)) and $2\alpha_i > \beta_i^2(\alpha_i - 1) + \beta_i$ (condition (2)) implies that $4\alpha_i > \beta_i^2(\alpha_i - 1) + \beta_i\alpha_i$, hence $\Delta x_{i,g}(k) > \Delta x_{i,g}(k+1)$ for $m \geq 6$.

QED

References

1. P. Dasgupta and R. Das. Dynamic pricing with limited competitor information in a multi-agent economy. In O. Eztion and P. Scheuermann, editors, *Proceedings of the 5th International Conference on Cooperative Information Systems (CoopIS). Lecture Notes in Computer Science*, volume 1906, pages 299–310, Berlin, September 2000. Springer-Verlag.
2. J. Morris DiMicco, A.R. Greenwald, and P. Maes. Dynamic pricing strategies under a finite time horizon. In *Proceedings of the 3rd ACM Conference on Electronic Commerce (EC '01)*, pages 95–104, New York, October 2001. ACM Press.
3. A.R. Greenwald and J.O. Kephart. Shopbots and pricebots. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 506–511, August 1999.
4. A.R. Greenwald and J.O. Kephart. Probabilistic pricebots. In *Proceeding of the 5th International Conference on Autonomous Agents*, pages 560–567, May 2001.
5. J.O. Kephart, J.E. Hanson, and A.R. Greenwald. Dynamic pricing by software agents. *Computer Networks*, 36(6):731–752, May 2000.
6. J. Morris. A simulation-based approach to dynamic pricing. Master's thesis, MIT, May 2001.
7. H.R. Varian. Pricing information goods. In *Proceedings of the Symposium on Scholarship in the New Information Environment.* Harvard Law School, May 1995.
8. H.R. Varian. Differential pricing and efficiency. *Firstmonday*, 1(2), August 1996. Available at http://www.firstmonday.dk/issues/issue2/different/.
9. G. Zacharia, T. Evgeniou, A. Moukas, P. Boufounos, and P. Maes. Economics of dynamic pricing in a reputation brokered agent mediated marketplace. In J. Liu and Y. Ye, editors, *E-Commerce Agents*, volume 2033 of *Lecture Notes in Artificial Intelligence*, pages 25–38, Berlin, 2001. Springer–Verlag.