André Berger, Alexander Grigoriev,
Joyce van Loon

Price Strategy Implementation

RM/08/035

JEL code: C44, C61

Maastricht research school of **E**conomics
of **TE**chnology and **OR**ganizations

Universiteit Maastricht
Faculty of Economics and Business Administration
P.O. Box 616
NL - 6200 MD Maastricht

phone : ++31 43 388 3830
fax : ++31 43 388 4873

Price Strategy Implementation

André Berger, Alexander Grigoriev, and Joyce van Loon*

Maastricht University, Quantitative Economics,
P.O.Box 616, NL–6200 MD Maastricht, The Netherlands
{a.berger,a.grigoriev,j.vanloon}@ke.unimaas.nl

Abstract. Consider a situation in which a company sells several different items to a set of customers. However, the company is not satisfied with the current pricing strategy and wishes to implement new prices for the items. Implementing these new prices in one single step might not be desirable, for example, because of the change in contract prices for the customers. Therefore, the company changes the prices gradually, such that the prices charged to a subset of the customers, the target market, do not differ too much from one period to the next. We propose a polynomial time algorithm to implement the new prices in the minimum number of time periods needed, given that the prices charged to the customers in the target market increase by at most a factor $1+\delta$, for predetermined $\delta > 0$. Furthermore, we address the problem to maximize the revenue when also a maximum number of time periods is predetermined. For this problem, we describe a dynamic program if the number of possible prices is limited, and a local search algorithm if all prices are allowed. Also, we present the integer program that models this problem. Finally, we apply the obtained algorithms in a practical study.

Keywords: Pricing problems, computational complexity, local search, integer linear program

1 Introduction

A company owns a set of different item types for sale, $K = \{1, \ldots, m\}$. We assume that every item type is available in unlimited supply. The set of potential customers is denoted by $J = \{1, \ldots, n\}$. Every customer $j \in J$ has a personal demand $d_{jk} \geq 0$ for every item type $k \in K$. The combination of the demand for all item types is called a customer's *contract*. Every customer $j \in J$ is single-minded and has a valuation b_j for the contract, which is the maximum amount she is willing to pay. Let $B = \max_{j \in J}\{b_j\}$ be an upper bound on the valuations.

In this paper, we regard the problem that the company faces when implementing a new price vector $p^* \in \mathbb{R}^m_+$. This price vector might be the optimal price vector obtained by solving the affine pricing algorithm as discussed in [8],

* Supported by METEOR, the Maastricht Research School of Economics of Technology and Organizations.

but can also be any other vector that the company wishes to implement. We assume that the current price vector is $p^0 \neq p^*$. Immediate introduction of the new prices in the next period (e.g. week, month) might not be desirable. For example, a sudden huge difference in prices may change the perception of customers or the position in the market in comparison to the competing companies. Therefore, the company wants to change prices *gradually*. Let $T = \{0, \ldots, t_{\max}\}$ be the set of time periods, where $t = 0$ is the current time and $t = t_{\max}$ is the end time of the implementation. For notational purposes, let $T^+ = T \setminus \{0\}$.

For every time step $t \in T$, the company has to set price vector $p^t = (p_1^t, \ldots, p_m^t) \in \mathbb{R}_+^m$. Given these prices, the contract price for customer $j \in J$ is an affine function on her personal demand

$$p^t(j) = d_{j0} + d_{j1}p_1^t + \cdots + d_{jm}p_m^t.$$

We call a customer a winner if she can afford her contract and receives it. We denote the set of winners by $W^t = \{j \in J : p^t(j) \leq b_j\}$. The company's revenue in time t is $\Pi^t = \sum_{j \in W^t} p^t(j)$. The company's total revenue is equal to

$$\Pi = \sum_{t \in T} \Pi^t = \sum_{t \in T} \sum_{j \in W^t} p^t(j).$$

Given price vector p^*, we can easily determine for which customers the price of the contract does not exceed valuation, that is, $W^* = \{j \in J : p^*(j) \leq b_j\}$. We call W^* the *target market*. When implementing the optimal pricing strategy, we have to satisfy the restriction that for these customers the price of the contract may increase at most a factor of $(1+\delta)$ between two consecutive periods for some positive δ, which will be predetermined by the company. That is, given price vectors p^0, p^* and δ, the company has to satisfy the following two constraints.

Constraint 1 $p^t(j) \leq (1+\delta)p^{t-1}(j), \qquad \forall\, j \in W^*,\, t \in T^+.$

Constraint 2 $p_k^{t_{\max}} = p_k^*, \qquad \forall\, k \in K.$

In this paper, we address two problems:

Problem 1. Given p^0, p^* and $\delta > 0$, find price vectors p^t for all $t \in T^+$, so as to minimize $t_{\max}$ subject to Constraints 1 and 2.

Problem 2. Given p^0, p^*, $\delta > 0$ and $t_{\max}$ that is large enough to satisfy Constraints 1 and 2, find price vectors p^t for all $t \in T^+$ subject to Constraints 1 and 2 so as to maximize $\sum_{t \in T} \Pi^t$.

1.1 Related work

The pricing regime referred to as *affine pricing*, as suggested by Grigoriev, van Loon and Uetz [8], appears to be particularly suitable for practical applications. It seems to be a more realistic model than the single item pricing model that was discussed in many previous papers on algorithmic profit maximizing pricing

problems [1, 3–10]. Note that single item pricing would require to determine a price for each copy of an item individually. In fact, affine pricing even generalizes the single item pricing models; see [8]. It is known that, for the affine pricing problem with a non-constant number of distinct items m, the maximum revenue is hard to approximate within a semi-logarithmic factor in the number of customers n [5]. However, for constant m, there exists a polynomial time algorithm [8].

1.2 Our results

For Problem 1, we show how to calculate the minimum number of time periods and we present an algorithm to find the price vectors with computation time $\mathrm{O}(nm + m \log B)$. For Problem 2 with integer price vectors, we describe a dynamic program that runs in polynomial time if the number of item types m is constant. For the problem with continuous price vectors, the dynamic program gives an approximate solution with performance guarantee $(1+\varepsilon)$ for any $\varepsilon > 0$, and a computation time of $\mathrm{O}(f(\varepsilon))$. Then, we introduce a local search algorithm that, in every time period $t > 0$, selects a pricing strategy that yields the highest total revenue to the company, but also satisfies the above constraints. Although we were not able to estimate the performance guarantee on the solution of this algorithm, computational studies show that the obtained revenue is not far from the optimal total revenue calculated by solving an integer linear program.

2 Implementation in minimal time

We assume that the price of any item type in the current solution is at least equal to some very small $\varepsilon > 0$, that is, $p^0_k \geq \varepsilon$ for all $k \in K$. Let q_j denote the ratio between customer j's contract price in the optimal and current pricing strategy and let q be the maximum ratio among all customers in the target market, that is,

$$q = \max_{j \in W^*} \{q_j\} = \max_{j \in W^*} \left\{ \frac{p^*(j)}{p^0(j)} \right\}.$$

Theorem 1. *Given current price vector p^0, the minimum number of steps to implement price vector p^* is $t_{\max} = t_q$, where $t_q = \max\{1, \lceil \log_{1+\delta} q \rceil\}$.*

Proof. Since $p^0 \neq p^*$, $t_{\max} \geq 1$, that is, we need at least one step to implement the new prices. By Constraint 1 we know that

$$p^*(j) \leq (1+\delta)^{t_{\max}} p^0(j), \quad \forall\, j \in W^*.$$

By definition of q, there exists some customer $j' \in W^*$ such that

$$q = \frac{p^*(j')}{p^0(j')} \leq (1+\delta)^{t_{\max}} \Leftrightarrow t_{\max} \geq \log_{1+\delta} q. \tag{1}$$

Combining inequality (1) with the facts that the minimum number of time periods needed is integer and at least equal to one, we have that $t_{\max} \geq t_q = \max\{1, \lceil \log_{1+\delta} q \rceil\}$.

Let the prices be located on the straight line in $\mathbb{R}^m_+$ between p^0 and p^*. For every price vector p^t, $t \in T$, we know that

$$p^t = \lambda^t p^* + (1 - \lambda^t) p^0, \tag{2}$$

for some $\lambda^t \in [0, 1]$. Now, we define

$$\lambda^t = \min_{j \in W^*} \left\{ \frac{(1+\delta)p^{t-1}(j) - p^0(j)}{p^*(j) - p^0(j)} \right\}. \tag{3}$$

Combining Equations (2) and (3), for every $t \in T^+$ and $j \in W^*$ we have

$$\begin{aligned} p^t(j) &= d_{j0} + d_{j1}\left((p^*_1 - p^0_1)\lambda^t + p^0_1\right) + \ldots + d_{jm}\left((p^*_m - p^0_m)\lambda^t + p^0_m\right) \\ &= p^0(j) + (p^*(j) - p^0(j))\lambda^t \\ &\leq p^0(j) + (p^*(j) - p^0(j))\frac{(1+\delta)p^{t-1}(j) - p^0(j)}{p^*(j) - p^0(j)} \\ &= (1+\delta)p^{t-1}(j), \end{aligned} \tag{4}$$

thus Condition 1 is satisfied. Moreover, the customer that minimizes λ^t also maximizes q, that is,

$$\begin{aligned} &\arg\min_{j \in W^*} \left\{ \frac{(1+\delta)p^{t-1}(j) - p^0(j)}{p^*(j) - p^0(j)} \right\} \\ &= \arg\min_{j \in W^*} \left\{ \frac{(1+\delta)\left(\left(p^*(j) - p^0(j)\right)\lambda^{t-1} + p^0(j)\right) - p^0(j)}{p^*(j) - p^0(j)} \right\} \\ &= \arg\min_{j \in W^*} \left\{ (1+\delta)\lambda^{t-1} + \frac{\delta p^0(j)}{p^*(j) - p^0(j)} \right\} = \arg\max_{j \in W^*} \left\{ \frac{p^*(j)}{p^0(j)} \right\}. \end{aligned} \tag{5}$$

Consequently, at step $t_q = \max\{1, \lceil \log_{1+\delta} q \rceil\}$ we reach p^*. Thus, $t_{\max} \leq t_q$. As we also showed that $t_{\max} \geq t_q$, the claim is proven. □

Using the steps in the proof above, we define the *straight line algorithm* as follows. Given $\delta > 0$ and price vectors p^0 and p^*, let $j' \in W^*$ be the customer with $q = p^*(j')/p^0(j')$. Then, combining Equations (3) and (5), we have

$$\lambda^t = \frac{(1+\delta)p^{t-1}(j') - p^0(j')}{p^*(j') - p^0(j')},$$

for all time steps $t = 1, \ldots, t_q$, where $t_q = \max\{1, \lceil \log_{1+\delta} q \rceil\}$. Let $p^t_k = (p^*_k - p^0_k)\lambda^t + p^0_k$ be the price of item type $k \in K$ in time period $t = 1, \ldots, t_q - 1$. Finally, in time period t_q, we implement price vector p^*.

Theorem 2. *The straight line algorithm finds a price implementation from p^0 to p^* in $O(nm + m \log B)$ time.*

Proof. The price implementation generated by the straight line algorithm satisfies Condition 1 by (4) and Condition 2 by definition of the final step in the algorithm. Thus, the algorithm solves Problem 1. Finding customer $j' \in W^*$ who maximizes q takes $O(nm)$ time, as we calculate the ratio between the contract prices for the new and current price vectors, for every customer in the target market. For all $t = 1, \ldots, t_q - 1$, where $t_q = \max\{1, \lceil \log_{1+\delta} q \rceil\}$, we need to find λ^t which can be done in $O(m)$ time, as we need to calculate $p^{t-1}(j')$. Then, implementing the price vector p^t takes $O(m)$ time, which results in a total running time of $O(nm + t_q m)$. Integer t_q is bounded from above by the logarithm of the largest valuation, as $q = \max_{j \in W^*}\{p^*(j)/p^0(j)\}$ only depends on the contract prices of the *winners* given price vector p^*. Consequently, the computation time of the straight line algorithm is $O(nm + m \log B)$. □

3 Maximize total revenue

First, for Problem 2 restricted to integer prices, we present a dynamic programming algorithm that runs in pseudo-polynomial time. Then, using straightforward scaling argument, we transform the algorithm to a $(1 + \varepsilon)$-approximation algorithm for the same problem with continuous prices, for any desired precision $\varepsilon > 0$.

In Problem 2 we want to find the strategy that maximizes the total revenue over $t_{\max}$ periods, that is,

$$\Pi = \sum_{t \in T} \Pi^t = \sum_{t \in T} \sum_{j \in W^t} p^t(j).$$

3.1 Dynamic program

We find the integer price vectors to optimize the total revenue by a pseudo-polynomial dynamic program $\mathcal{A}_2$, which determines the longest $u - v$ path in an acyclic digraph. Let $t_{\max}$ be the number of time periods the company wants to use to implement pricing strategy p^*, and let $T = \{0, 1, \ldots, t_{\max}\}$. We construct a digraph $D = (N, A)$ of $t_{\max} + 1$ layers, in which we find the longest $u - v$ path. A node is of the form (t, p^t), where t is the time period, or layer, and p^t is an integer price vector. Arcs are only created between two consecutive layers.

In layer 0, there is only node $u = (0, p^0)$, and in layer $t_{\max}$ there is only node $v = (t_{\max}, p^{t_{\max}})$, where $p^{t_{\max}} = p^*$. There exists an arc between node $(t-1, p^{t-1})$ and node (t, p^t) if

- $t \in T^+$,
- p^{t-1} and p^t are integer price vectors,
- $p^t(j) \leq (1 + \delta)p^{t-1}(j)$ for all $j \in W^*$.

The length of the arc is Π^{t-1}. The longest $u - v$ path corresponds to the pricing strategy that yields maximum total revenue. Note that the length of this path is equal to the total revenue.

Theorem 3. *Dynamic programming algorithm $\mathcal{A}_2$ solves Problem 2 with integer price vectors in time $O(t_{\max}^2 B^{2m} n)$.*

Proof. Correctness of the dynamic program follows from the fact that we enumerate over all possible integer price vectors. The number of nodes in layer t is $\mathrm{O}(B^m)$, which means that in total we have $\mathrm{O}(t_{\max} B^m)$ nodes in the digraph. The number of arcs is $\mathrm{O}(t_{\max}^2 B^{2m})$. Determining the length of an arc, that is, Π^{t-1}, takes $\mathrm{O}(nm)$ time. Finding the longest $u - v$ path in a digraph is linear in the number of arcs [2], in which we include the calculation of the arc lengths. So the claimed complexity follows. □

In order to solve the problem with continuous price vectors, let $\varepsilon > 0$ be the precision of the prices. Allowing prices to be integer multipliers of ε, we arrive at the following corollary.

Corollary 1. *Dynamic programming algorithm $\mathcal{A}_2$ is a $(1+\varepsilon)$-approximation algorithm running in $O(t_{\max}^2 (B/\varepsilon)^{2m} n)$, for any $\varepsilon > 0$.*

Because of this corollary and the fact that the running time is fairly large, we introduce a local search algorithm.

3.2 Local search

The local search algorithm that we present in this section executes in every time period $t \in T^+$ the affine pricing algorithm; see [8, Algorithm 1]. The local search algorithm $\mathcal{A}_3$ is formally described in Algorithm 1. Afterwards, we present an example. In this section, we assume that the price vector p^* is the optimal price vector for the given instance of the affine pricing problem.

Note that this algorithm assures that as soon as we are able to reach the optimal price vector p^*, we will stay at this level as there is obviously no better price vector in reach.

Example 1. Consider a setting with three customers and two items, and $\delta = 1$. The valuation constraints $p(j) \leq b_j$ are $p(1) = 16p_1 + 32p_2 \leq 512 = b_1$, $p(2) = 20p_1 + 20p_2 \leq 400 = b_2$ and $p(3) = 28p_1 + 16p_2 \leq 448 = b_3$, displayed in Figure 1. Using the affine pricing algorithm, we know that the optimal pricing is $p^* = (8, 12)$. The current price vector is $p^0 = (2, 1)$. With this current price vector, customer 1 pays 64, customer 2 pays 60 and customer 3 pays 72. First, we create an arrangement of linear inequalities

$$\begin{aligned} 16\, p_1^1 + 32\, p_2^1 &\leq (1+\delta)64 = 128 \\ 20\, p_1^1 + 20\, p_2^1 &\leq (1+\delta)60 = 120 \\ 28\, p_1^1 + 16\, p_2^1 &\leq (1+\delta)72 = 144 \\ p_1^1 &\geq 0 \\ p_2^1 &\geq 0. \end{aligned}$$

These constraints are represented by the dotted lines in the figure. The optimal price vector is $p^1 = (4, 2)$. Repeating this procedure leads to $p^2 = (8, 4)$. Then,

Algorithm 1: Local search algorithm $\mathcal{A}_3$

Given p^0 and $\delta > 0$. Let $\Pi^* = \Pi^0$;
foreach $t \in T^+$ **do**
  Let $\Pi^t = 0$, $p^t = 0$ and $W^t = J$;
  foreach *set of* m *linearly independent equalities out of the* $|W^*| + m$ *equalities* $p^t(j) = (1+\delta)p^{t-1}(j)$, $j \in W^*$, *and* $p^t_k = 0$, $k \in K$ **do**
    Determine price vector p that is characterized by these m equalities;
    Let $p(j) = d_{j0} + d_{j1}p_1 + \cdots + d_{jm}p_m$ be the price of the contract requested by customer $j \in J$;
    Let $W = \{j \in J : p(j) \leq b_j\}$ be the set of winners;
    Let $\Pi = \sum_{j \in W} p(j)$ be the total revenue;
    if $\Pi > \Pi^t$ **then**
      Let $\Pi^t = \Pi$, $p^t = p$ and $W^t = W$;
    end
  end
  $\Pi^* += \Pi^t$;
end

we get $16p_1^2 + 32p_2^2 \leq (1+\delta)256 = 512$, $20p_1^2 + 20p_2^2 \leq (1+\delta)240 = 480$, and $28p_1^2 + 16p_2^2 \leq (1+\delta)288 = 576$. Thus, this gives the arrangement defined by the valuation constraint of customer 1, and the dash-dotted lines in the figure. The optimal solution p^* is reachable, so the algorithm selects this price vector and keeps selecting it for all periods $t \geq 3$, $t \in T$.

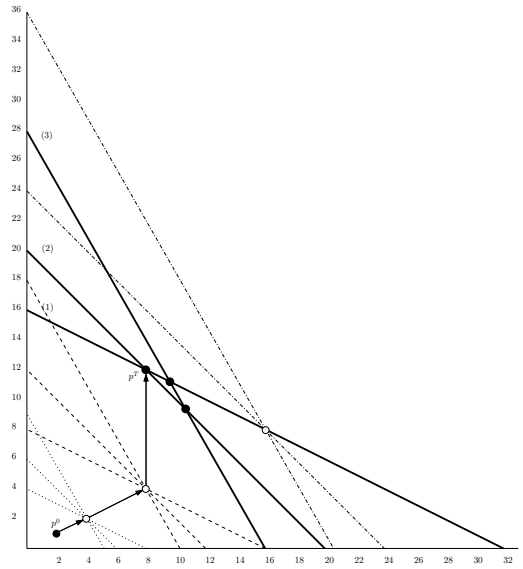

Fig. 1. Graphical representation of an instance for $m = 2$.

Theorem 4. *Local search algorithm $\mathcal{A}_3$ runs in $O(t_{\max}(n+m)^m(m^3+nm))$ time, which is in $O(t_{\max}n^{m+1})$ if the number of item types is constant.*

Proof. In the affine pricing algorithm that is applied during the local search, we consider $\binom{|W^*|+m}{m} \in \mathrm{O}((n+m)^m)$ systems of m equalities each. In each of these iterations, we solve a linear system in m variables and m constraints to determine the price vector, which takes $\mathrm{O}(m^3)$ time. Computation of the contract prices, winners, and the revenue takes $\mathrm{O}(nm)$ time. We perform these steps for all $t \in T^+$, so the claimed complexity follows. □

4 Computational experiments

We apply the straight line algorithm $\mathcal{A}_1$ and the local search algorithm $\mathcal{A}_3$ to implement the optimal pricing strategy for a telephone operator. The item types are available in unlimited supply, as we are selling digital goods. We assume customers to be single-minded, that is, each customer either accepts the offer for the contract or she leaves to a competitor. In this practical application, we determine the start-up tariff, a price per minute for calling within the country and abroad, and a price per sent text message. The data we use contain detailed information about the phone usage of many customers. For each customer, we can exactly determine the demand for the different item types and therefore the contract she requests. Let p_1 be the start-up price for a call, p_2 the price per minute for calling domestically, p_3 the price per text message, and p_4 the price per minute for calling abroad. Then, the price for the contract of customer $j \in J$ is defined as

$$p(j) = d_{j1}p_1 + d_{j2}p_2 + d_{j3}p_3 + d_{j4}p_4,$$

where d_{j1} is the number of calls customer j wants to make, d_{j2} is the number of minutes she calls within the country, d_{j3} is the number of text messages she sends, and d_{j4} is the number of minutes she calls abroad. We assume that customers are rational, and therefore select the cheapest offer for their contract. Thus, a customer only accepts our offer for her contract if we offer the cheapest price in the market. Therefore, we define the *valuation* of a customer the cheapest price for her contract at any competitor.

We apply the described algorithms to three different samples of a data set that resembles reality. Every sample contains ten customers. The current price vector in euro is $p^0 = (0.01, 0.06, 0.01, 0.75)$ and $\delta = 0.05$. In this section we determine the minimum number of time periods, and the number of periods used by local search algorithm $\mathcal{A}_3$ to implement the optimal prices. Then, we predetermine $t_{\max} = 40$ and find the optimal pricing strategy. This strategy is calculated by the integer program described below. In an optimal solution at time $t \in T$, we know that variable $\pi_j^t = p^t(j)$ if $j \in W^t$ and $\pi_j^t = b_j$ otherwise. Binary variable x_j^t is equal to 0 if $j \in W^t$ and 1 otherwise.

$$
\begin{array}{lll}
\max & \sum_{t=0}^{T} \sum_{j \in J} \left(\pi_j^t - x_j^t b_j\right) & \\
\text{s.t.} & p^t(j) \leq (1+\delta) p^{t-1}(j) & \forall\, t \in T^+,\, \forall\, j \in W^* \\
& \pi_j^t \leq b_j & \forall\, t \in T,\, \forall\, j \in J \\
& \pi_j^t \leq p^t(j) & \forall\, t \in T,\, \forall\, j \in J \\
& p^t(j) \leq (1+\delta) B x_j^t + b_j & \forall\, t \in T,\, \forall\, j \in J \\
& p_k^{t_{\max}} = p_k^* & \forall\, k \in K \\
& p_k^t \geq 0 & \forall\, t \in T,\, \forall\, k \in K \\
& \pi_j^t \geq 0 & \forall\, t \in T,\, \forall\, j \in J \\
& x_j^t \in \{0,1\} & \forall\, t \in T,\, \forall\, j \in J
\end{array}
$$

Sample 1 For the first sample, the optimal price vector is $p^* = (0.175136, 0.170037, 0.087114, 0.766734)$. The minimum number of time steps needed to implement p^* starting from p^0 is 31. The total revenue generated by the straight line algorithm is 64230.64. The optimal revenue for implementing p^* in 31 steps is 70936.90, which means that the revenue of the straight line algorithm is 90.55% of the optimal revenue. The local search algorithm needs at least 34 steps to implement p^*. However, running this algorithm for 31 steps generates a total revenue of 70968.47, which is even larger than the optimal revenue when we do implement p^*.

For the second problem, we set $t_{\max} = 40$. The optimal revenue is equal to 99020.27, and the optimal price vector is implemented from step 33 onwards. The local search algorithm yields a revenue of 98898.67, which is 99.88% of the optimal revenue. The straight line algorithm outputs a revenue of 92192.83, which is 93.11% of optimal. Both these algorithms instantly give the optimal solution. Solving the integer program takes 2 seconds.

Sample 2 The optimal price vector for the second sample is $p^* = (0.109423, 0.173030, 0.216197, 0.666756)$. The minimum number of time steps needed to implement p^* starting from p^0 is 29. The total revenue generated by the straight line algorithm is 35339.58. The optimal revenue for implementing p^* in 29 steps is 37449.57, which means that the revenue of the straight line algorithm is 94.37% of the optimal revenue. The local search algorithm needs at least 32 steps to implement p^*. Running this algorithm for 29 steps generates a total revenue of 37244.64, which is 99.45% of the optimal revenue when we do implement p^*.

For the second problem, we set $t_{\max} = 40$. The optimal revenue is equal to 56229.31, and the optimal price vector is implemented from step 29 onwards. The local search algorithm yields a revenue of 56004.24, which is 99.60% of the optimal revenue. The straight line algorithm outputs a revenue of 54121.09, which is 96.25% of optimal. Both these algorithms instantly give the optimal solution. Solving the integer program takes 338 seconds.

Sample 3 For the last sample, the optimal price vector is $p^* = (0.112089, 0.188362, 0.102176, 1.467432)$. The minimum number of time steps needed to implement p^* starting from p^0 is 33. The total revenue generated by the straight line algorithm is 24293.19. The optimal revenue for implementing p^* in 33 steps is 27825.56, which means that the revenue of the straight line algorithm is 87.31%

of the optimal revenue. In this sample, the local search algorithm also needs 33 steps to implement p^*, and generates a revenue of 27736.29, which is 99.68% of the optimal revenue.

For the second problem, we set $t_{\max} = 40$. The optimal revenue is equal to 36833.93, and the optimal price vector is implemented from step 33 onwards. The local search algorithm yields a revenue of 36744.66, which is 99.76% of the optimal revenue. The straight line algorithm outputs a revenue of 33301.56, which is 90.41% of optimal. Both these algorithms instantly give the optimal solution. Solving the integer program takes 3 seconds.

	Sample 1		Sample 2		Sample 3	
min $t_{\max}$	31		29		33	
MIP	70936.90		37449.57		27825.56	
Straight	64230.64	90.55%	35339.58	94.37%	24293.19	87.31%
LS till $t_{\max}$	70968.47	100.04%	37244.64	99.45%	27736.29	99.68%
$t_{\max}$	40		40		40	
MIP	99020.27		56229.31		36833.93	
Straight	92192.83	93.11%	54121.09	96.25%	33301.56	90.41%
LS	98898.67	99.88%	56004.24	99.60%	36744.66	99.76%

Table 1. Summary of computational results.

5 Conclusion

Regarding Problem 1, we can easily calculate the minimum $t_{\max}$, and find price vectors p^t for all $t \in T^+$ in polynomial time using the straight line algorithm. The local search algorithm might need more steps to implement p^*, but using this algorithm can be more profitable to the company than even using the integer program, as there we have the requirement that $p^{t_{\max}}$ should be equal to p^*.

For Problem 2 restricted to integer prices, we present a dynamic programming algorithm that runs in pseudo-polynomial time. We extend this result and derive a $(1+\varepsilon)$-approximation algorithm for the same problem with continuous prices, for any desired precision $\varepsilon > 0$. The local search algorithm seems very useful in practice, and yields close-to-optimal revenues according to our computational results.

References

1. G. Aggarwal, T. Feder, R. Motwani, and A. Zhu, *Algorithms for multi-product pricing*, Automata, Languages and Programming - ICALP 2004 (J. Díaz, J. Karhumäki, A. Lepistö, and D. Sannella, eds.), Lecture Notes in Computer Science, vol. 3142, Springer, 2004, pp. 72–83.

2. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows*, Prentice Hall, New Jersey, 1993.
3. M. F. Balcan and A. Blum, *Approximation algorithms and online mechanisms for item pricing*, Proceedings of the 7th ACM Conference on Electronic Commerce, ACM, 2006, pp. 29–35.
4. P. Briest and P. Krysta, *Single-minded unlimited supply pricing on sparse instances*, Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms, ACM-SIAM, 2006, pp. 1093–1102.
5. E. D. Demaine, U. Feige, M.T. Hajiaghayi, and M. R. Salavatipour, *Combination can be hard: Approximability of the unique coverage problem*, Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms, ACM-SIAM, 2006, pp. 162–171.
6. K. Elbassioni, R. Sitters, and Y. Zhang, *A quasi-ptas for profit-maximizing pricing on line graphs*, Proceedings of the 15th Annual European Symposium on Algorithms (L. Arge and E. Welzl, eds.), Lecture Notes in Computer Science, vol. 4698, Springer, 2007, pp. 451–462.
7. A. Grigoriev, J. van Loon, R. Sitters, and M. Uetz, *Optimal pricing of capacitated networks*, Networks (2008, to appear).
8. A. Grigoriev, J. van Loon, M. Sviridenko, M. Uetz, and T. Vredeveld, *Bundle pricing with comparable items*, Algorithms - ESA 2007 (L. Arge, M. Hoffmann, and E. Welzl, eds.), Lecture Notes in Computer Science, vol. 4698, Springer, 2007, pp. 475–486.
9. V. Guruswami, J. D. Hartline, A. R. Karlin, D. Kempe, C. Kenyon, and F. McSherry, *On profit-maximizing envy-free pricing*, Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, ACM-SIAM, 2005, pp. 1164–1173.
10. J. D. Hartline and V. Koltun, *Near-optimal pricing in near-linear time*, Algorithms and Data Structures - WADS 2005 (F. K. H. A. Dehne, A. López-Ortiz, and J.-R. Sack, eds.), Lecture Notes in Computer Sciences, vol. 3608, Springer, 2005, pp. 422–431.