Strategies for Maximizing Seller's Profit under Unknown Buyer's Valuations

Bella Belegradek[1,3], Konstantinos Kalpakis[1], and Yelena Yesha[1,2,3]

Abstract

Suppose there is a seller that has an unlimited number of units of a single product for sale. The seller at each moment of time posts a price for his/her product. Based of the posted price, at each moment of time, a buyer decides whether or not to buy a unit of that product from the seller. The only information about the buyer available to the seller is the seller's sales history. Further, we assume that the maximal unit price the buyer is willing to pay does not change over time. The question then is how should the seller price his/her product to maximize profits?

To address this question, we use the notion of *loss functions*. Intuitively, a loss function is a measure, at each moment of time, of the lost opportunity to make a profit. In particular, we provide a polynomial–time algorithm that finds a pricing algorithm (strategy) for the seller that minimizes the average (total) losses over time. Further, we provide efficient algorithms for finding pricing strategies for maximizing the cumulative seller's profits when the sales period is limited, there is a limited supply of the product for sale, and the buyer's valuation is a non–increasing function of time, where those functions are known to the seller. We also present preliminary results on pricing strategies that minimize the maximum loss, and we show show that there is no strategy minimizing both the total loss and the maximum loss at the same time.

Keywords: *Pricing strategies, market models, electronic commerce, computational learning theory, dynamic programming, search trees, discrete algorithms.*

1 Introduction

We present algorithms for finding pricing strategies to maximize profits of a seller in the business-to-customer catalog sales environment, one of the major modes of Electronic Com-

[1]Computer Science & Electrical Engineering Department, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250. E-mail: {bbeleg1,kalpakis,yeyesha}@cs.umbc.edu

[2]Center of Excellence in Space Data and Information Sciences, NASA Goddard Flight Space Center, Greenbelt, MD 20771.

[3]Supported in part by the Center for Advanced Studies, IBM Toronto Laboratory.

merce. This electronic commerce environment has two important features that motivated the model and algorithms presented in this paper:

- The anonymity of potential buyers and, in some cases, their unwillingness to disclose any information about themselves.
- The ability of a seller to make quick changes of the prices or even of the entire pricing policy. Such an ability is based on the natural use of computers on the seller's side not only for storage and retrieval of catalog data, but also for collecting and analyzing the implicit information about the buyers and the situation on the market, whence incorporating this information into dynamic pricing algorithms. On the other hand, the electronic form of sales makes the distribution of the updated prices fast and almost costless.

We use the following formal model which corresponds to a single seller perception of the market. Time is divided into periods (of possible different lengths) indexed by non-negative integers. The seller has a single kind of a product for sale. At the beginning of every period the seller posts the current price for the product and, perhaps, the number of units available for sale in that period. At every period, there is a number of buyers with possible different maximum prices each one of them is willing to pay per unit of the product. A buyer always buys a unit of the product when the seller's price does not exceed the buyer's maximum price. We assume no strategic behavior for the buyers. The only information available to the seller is the history of previous sales. The history of sales from the point of view of the seller is the sequence of prices set for each prior period and quantities of the product sold.

The main question addressed in this paper is "what is the best pricing strategy for the seller?"

In order to simplify the discussion and, since the seller never sells below cost, we will refer to the difference between the maximum price the buyer is willing to pay for a unit and the cost of a unit of a product to the seller as the *buyer's valuation*. Also the "seller's price" will mean the difference between the sale price set for a given period and the seller's cost per unit at that period rather than the sale price itself.

In this paper we use a restricted version of the model above, namely:

- At every time period t, there is one buyer in the market.
- At each time t, at most one unit of the product can be sold.
- The buyer's valuation X_0 at time $t = 0$ is assumed to be within a given range $[0, \ldots, N]$. The value of X_0 is unknown to the seller, but the valuation change functions, which will be formally defined in section 5, are deterministic and known to the seller in advance.

Similar models are well known in economics of monopolistic pricing (see [1], [4], [9], [11], [12]). Although they consider markets with a single seller and multiple buyers, they focus on the comparison of different sales mechanisms such as auctions, different prices set for different buyer groups or single set price ([9]); or they study equilibria for their respective economic models. [4] Ausabel and Deneckere [1], and Sobel [12] prove that there exist equilibria for all levels of seller profits, if the buyers have an access to the sales history and are capable of strategic behavior. Benabou [4] studies equilibria and optimal pricing strategies for the seller in inflationary monopolistic economics, where he assumes that buyers have strategic behavior and have knowledge of the sales history.

Awerbuch and Azar [2], and El–Yaniv et al. [8] use competitive analysis of algorithms to solve optimization problems arising from economic problems. Competitive analysis of algorithms produces too coarse results to be used in practice, since it is primarily concerned with worst–case analysis. Learning theory seems to be more useful for these problems, especially when there are no time bounds, i.e. in optimization problems with the infinite horizon.

We provide computationally effective algorithms for finding strategies for dynamic price changes that maximize the profits for the seller. We use the notion of *loss functions* from Learning Theory to find an optimal pricing strategy for the seller. Intuitively, a loss function is a measure, at each moment of time, of the lost opportunity to make a profit. It is clear then that in order to maximize profits, one should minimize some loss function over time.

We provide a polynomial–time algorithm that finds a pricing algorithm (strategy) for the seller that minimizes the total (average) losses over time. Further, we present preliminary results on pricing strategies that minimize the maximum possible loss. We also show that there is no strategy minimizing both the total loss and the maximum loss at the same time. Moreover, we provide efficient algorithms for finding pricing strategies for maximizing cumulative seller's profit when the sales period is limited, there is a limited supply of the product for sale, and the buyer's valuation is a non–increasing function of time (those functions are known to the seller).

The rest of this paper is organized as follows. In section 2 we provide preliminaries. In section 3, we provide an algorithm for computing a pricing strategy for minimizing the total loss. In section 4, we provide lower and upper bounds on the maximum loss. In Section 5 we present several algorithms that explicitly maximize the total profits when the supply or sales period are limited. Section 6 concludes this paper.

[4] An equilibrium is a stable state in an economic model, such that a deviation from this state leads to a loss for the deviating side.

2 Preliminaries

Our goal is to find an optimal seller's pricing algorithm for a given loss function and an interval of possible buyer's valuations (or *maximum buyer's prices*). We assume that the maximum buyer's price reflects not the actual price, but rather an additional amount the buyer is willing to pay over a fixed minimum seller's price. Further, unless we state otherwise, a buyer's valuation is constant.

The notion of "maximizing profit" is a little tricky when the time to conduct the sales is unlimited, as it often happens in many problems associated with on-line algorithms in general and on-line learning theory in particular. Indeed, there is no dominating algorithm that can provide the maximum achievable profit for any possible buyer's price.

The straightforward worst–case analysis does not work well either. The maximum profit in the worst case is 0, and it does not depend on the pricing strategy. The worst case profit is always achieved for the lowest possible price and the best performing algorithm is the most conservative one, the one which starts with the lowest possible price and keeps increasing it by 1 as long as a deal occurs. After the first time a deal does not occur, the seller returns to the previous price which is the true buyer's valuation. It is easy to see that this algorithm does not provide us with maximum profit when the buyer's valuation is close to its upper bound.

We use *loss functions* (frequently used in on-line learning theory), which provide another way to estimate the "goodness" of an algorithm, that unifies both average–case analysis and a sensible worst–case performance of that algorithm. The typical setting for an on-line learning problem includes a sequence of learner's trials and teacher's (or just the outer world's) responses, which are often the only input to a learning algorithm to learn an unknown concept or distribution [4, 13]. The closeness of learner's perception of the target concept to the target concept itself is measured by a loss function. Then, the best learning algorithm is the one that minimizes the value of that loss function over the space of all possible inputs.

Two types of loss functions are commonly used: (a) those in which the total loss of a sequence of trials is equal to the number of negative responses when all the responses are binary [3, 6], and (b) those that are based on the distance, in some metric space, between the outer world sample and the estimate of this sample, predicted by the learning algorithm. In the latter case, the learner usually gets the sample itself as the teacher's response [10, 13]. We use hybrid loss functions.

Let $[0, \ldots, N]$ be the interval of the possible buyer's prices, X be the true maximum buyer's price, let $\mathcal{A}$ be the pricing algorithm to be used by the seller, and let P_t be the unit price at time t set by $\mathcal{A}$. The *loss of* $\mathcal{A}$ *at time* t is the difference between the maximum profit the seller could get at time t (which is equal to X) and the profit actually obtained at

time t:

$$loss_{\mathcal{A}}(X,t) = \begin{cases} X, \text{ if } P_t > X, \\ X - P_t, \text{ otherwise.} \end{cases} \tag{1}$$

The *loss of* $\mathcal{A}$ *on input* X is

$$loss_{\mathcal{A}}(X) = \sum_{t=0}^{\infty} loss_{\mathcal{A}}(X,t) \tag{2}$$

We consider the following two loss functions:

(a) The *total loss* of the pricing algorithm $\mathcal{A}$:
$total\_loss_{\mathcal{A}} = \sum_{X=0}^{N} loss_{\mathcal{A}}(X)$.

(b) The *maximum loss* of the pricing algorithm $\mathcal{A}$:
$max\_loss_{\mathcal{A}} = \max_{0 \le X \le N} loss_{\mathcal{A}}(X)$.

Note that $total\_loss_{\mathcal{A}}$ is the average–case loss multiplied by N when X is uniformly distributed on $[0, N]$, and that $max\_loss_{\mathcal{A}}$ is the worst–case loss.

Obviously, if the seller knows X at time τ, then, by setting $P_t = X$ for each $t \ge \tau$, the seller will not incur any losses after time τ. Therefore, if a pricing algorithm $\mathcal{A}$ is able to recognize any possible maximum buyer's price X by a constant time t', then $loss_{\mathcal{A}}(X)$ is finite for every $X \in [0, \ldots, N]$. Thus, in order for a pricing algorithm to be optimal with respect to any of the two loss functions defined above, such an algorithm must be able to find the maximum buyer's price in a finite amount of time and must provide finite losses on all possible inputs in a given range.

Let $\mathcal{P}_t(X)$ be the set of possible maximum buyer's prices at time t according to $\mathcal{A}$, when the true maximum buyer's price is X. Clearly, $\mathcal{P}_t(X)$ depends on X and the prices set by $\mathcal{A}$ until time t. Further, $\mathcal{P}_0(X) = [0, \ldots, N]$ and $X \in \mathcal{P}_t(X)$ for any $0 \le X \le N$ and for all $t \ge 0$. We call a pricing algorithm $\mathcal{A}$ *admissible*, if either $\mathcal{P}_t(X) = \{X\}$ or $\mathcal{P}_{t+1}(X) \subset \mathcal{P}_t(X)$ and $\mathcal{P}_{t+1}(X) \ne \mathcal{P}_t(X)$ for all $X \in [0, \ldots, N]$ and all $t \ge 0$. Intuitively, an admissible pricing algorithm does not ask unnecessary questions, thus it makes progress towards finding the maximum buyer's price at every time step. $\mathcal{A}$ learns (knows or finds) the maximum buyer's price X at time τ if $\mathcal{P}_\tau(X) = \{X\}$.

Lemma 2.1 *Let* $\mathcal{A}$ *is an admissible pricing algorithm for an interval of buyer's prices* $[0, \ldots, N]$. *Then,* $\mathcal{A}$ *can be represented as a binary search tree with* $N+1$ *leaves and with* N *internal nodes of degree* 2. *Furthermore, there exist admissible optimal pricing algorithms for both the total loss and the maximum loss functions.*

Proof: Observe that $\mathcal{A}$, by setting the price to P_t at time t, effectively asks the question "Is $P_t \leq X$?". The occurrence of a deal at time t is equivalent to an affirmative answer to this question. Thus, if $\mathcal{P}_t(X) \neq \{X\}$ then

$$\mathcal{P}_{t+1}(X) = \begin{cases} \{x \ : \ x \in \mathcal{P}_t(X) \text{ and } x \geq P_t\}, \text{ if } P_t \leq X, \\ \\ \{x \ : \ x \in \mathcal{P}_t(X) \ \text{ and } x < P_t\}, \text{ if } P_t > X, \end{cases} \tag{3}$$

Since the occurrence or non–occurrence of a deal is the only new information made available to $\mathcal{A}$ at every t, $\mathcal{A}$ can be represented by a binary search tree T for the interval $[0, \ldots, N]$. Every question by $\mathcal{A}$ corresponds to an internal node of degree 2 in T and every leaf of T corresponds to a possible value of X.

Consider now a pricing algorithm $\mathcal{A}'$ that finds the buyer's valuation in finite time. If for some X the set of possible values of X, as it is viewed by $\mathcal{A}'$, is the same for several consecutive moments of time $t, t+1, \ldots, t+k$ and reduces at time $t+k+1$, then, the algorithm $\mathcal{A}''$, that "skips" all the (redundant) steps of $\mathcal{A}'$ from time t until time $t+k$ but is otherwise identical to $\mathcal{A}'$, does not increase the losses for any X. Thus, with respect to any of our loss functions, $\mathcal{A}''$ is not worse than $\mathcal{A}'$. Repeating this procedure for a finite number of times, we can obtain an admissible algorithm that provides at most the same values for the both loss functions as $\mathcal{A}'$ does. Therefore, there always exists an optimal admissible algorithm for any of our loss functions.

■

By Lemma 2.1, we only need to consider admissible pricing algorithms when searching for optimal pricing algorithms. Furthermore, since to any admissible pricing algorithm there corresponds a binary search tree with $N+1$ leaves and with N internal nodes of degree 2, one could find an optimal pricing algorithm by exhaustive search over all such binary search trees. However, since there are $\dfrac{1}{1+N}\binom{2N}{N}$ such search trees, this approach is computationally infeasible. A more efficient search method is needed in order to find optimal admissible pricing algorithms.

Let T be the binary search tree representation of an admissible pricing algorithm $\mathcal{A}$ for $[0, \ldots, N]$. We cast the definition of loss functions in terms of T. Consider a (unique) search path in T from its root to the leaf labeled X. We denote this path by $y_0, y_1, \ldots, y_k$, where $y_0 = root(T)$ and y_k is the leaf X. The loss of the search for X on the edge (y_i, y_{i+1}) or, alternatively, the loss of the search for X at the i-th step is

$$loss_T(X, (y_i, y_{i+1})) = \begin{cases} X, \text{ if } X < y_i \\ X - y_i, \text{ otherwise,} \end{cases} \tag{4}$$

and the loss of the search for X in T is

$$loss_T(X) = \sum_{i=0}^{k-1} loss_T(X, (y_i, y_{i+1})). \tag{5}$$

The total loss $total\_loss_T(N)$ and maximum loss $max\_loss_T(N)$ are defined as the sum and the maximum, respectively, of the individual losses for all $X \in [0, \ldots, N]$. The goal is then the following: given the interval $[0, \ldots, N]$, construct a binary search tree T_{total} (T_{max}) that minimizes the total (maximum) loss among all possible binary search trees.

3 Minimizing the Total Loss

First, we compute the value of the total loss for the most natural search tree – the (perfectly) balanced binary search tree. Although, as we prove later in this section, the balanced binary search tree does not minimize the total loss, the obtained value is useful as an upper bound on the total loss.

Lemma 3.1 *The total loss of the (perfectly) balanced binary search tree for the interval* $[0, \ldots, 2^k - 1]$, $k \geq 0$, *is equal to* $k(4^{k-1} - 2^{k-1})$.

Proof: Consider a (perfectly) balanced binary search tree T for the interval $[0, \ldots, 2^k - 1]$. The root of T is 2^{k-1}. Let $x_0 x_1 \ldots x_{k-1}$ denote the k–digit binary representation of an integer $X \in [0, \ldots, 2^k - 1]$. Let $flip(i, X)$ be a function that complements (flips) the i-th bit in the k–digit binary representation of X. Let X_0 be a number in $[0, \ldots, 2^{k-1} - 1]$. Every number in this interval has its 0-th bit (most significant bit) equal to 0. Let $X_1 = flip(0, X_0)$. We express $loss(X_1)$ as a function of $loss(X_0)$. Recall that all the bits in X_0 and X_1, except for the most–significant bit (0-th bit), are identical, and so let x_i denote the i-th bit in both numbers, for any $1 \leq i \leq k-1$. Let $loss_i(X)$ denote the loss of the search for X at the i-th step in T, $0 \leq i \leq k-1$. Then,

$$loss_i(X_1) = \begin{cases} loss_i(X_0), & \text{if } i = 0 \\ loss_i(X_0), & \text{if } i \geq 1 \text{ and } x_i = 1 \\ loss_i(X_0) + 2^{k-1}, & \text{if } i \geq 1 \text{ and } x_i = 0 \end{cases} \tag{6}$$

and

$$loss(X_1) = loss(X_0) + 2^{k-1} \sum_{i=1}^{k-1} (1 - x_i). \tag{7}$$

Further, the $loss(X_0)$ in T with root 2^{k-1} is equal to $loss_{T_L}(X_0) + X_0$, where T_L denotes the left subtree of the root of T. Thus,

$$\begin{aligned} total\_loss(T) &= \sum_{X=0}^{2^{k-1}-1} loss_T(X) + \sum_{X=0}^{2^{k-1}-1} loss_T(flip(0,X)) = \\ &= 2\left(\sum_{X=0}^{2^{k-1}-1} loss_{T_L}(X) + \sum_{X=0}^{2^{k-1}-1} X\right) + 2^{k-1}\sum_{X=0}^{2^{k-1}-1}\sum_{i=1}^{k-1}(1-x_i) = \\ &= 2 \cdot total\_loss(T_L) + 2^{2k-3}k - 2^{k-2}. \end{aligned} \tag{8}$$

Equation (8) gives us the recurrence: $f(k) = 2 \cdot f(k-1) + 2^{2k-3}k - 2^{k-2}$, with $f(1) = 0$ and $total\_loss(T) = f(k)$. The solution of this recurrence is $f(k) = k(4^{k-1} - 2^{k-1})$.

■

Second, using dynamic programming [7], we provide a polynomial–time algorithm that finds an admissible pricing algorithm that minimizes the total loss.

Theorem 3.2 *Given a positive integer N, we can find an admissible algorithm that minimizes the total loss on $[0, \ldots, N]$, in $O(N^3)$ time while using $O(N^2)$ space.*

Proof: Let T be a binary search tree for an interval $[i, \ldots, j]$ which provides the minimum total loss for this interval. Let k be the root of T. Since T is the representation of an algorithm that reduces the range of possible values of X on every step, we conclude that $i + 1 \leq k \leq j$.

Observe that for any $X < k$, $loss_T(X) = loss_{T_L}(X) + X$ and that for any $Y \geq k$, $loss_T(Y) = loss_{T_R}(Y) + Y - k$, where T_L and T_R are the left and right subtrees of the root of T respectively. Therefore,

$$\begin{aligned} total\_loss_T(i,j) &= \sum_{X=i}^{k-1}(loss_{T_L}(X) + X) + \sum_{Y=k}^{j}(loss_{T_R}(Y) + Y - k) = \\ &= total\_loss_{T_L}(i,k-1) + total\_loss_{T_R}(k,j) + \frac{(j-i+1)(j+i)}{2} \\ &\quad -(j-k+1)k. \end{aligned} \tag{9}$$

Clearly, from equation (9), we can see that the problem has the optimal sub–structure property [7], *i.e.* the restriction of the optimal solution to any subproblem is the optimal solution of this subproblem.

Let $TL(i,j)$ denote the minimum total loss on $[i,\ldots,j]$. Then, from equation (9) we get the following recurrence

$$TL(i,j) = \min_{i+1 \le k \le j} \{TL(i,k-1) + TL(k,j) + (j-i+1)(j+i)/2 - (j-k+1)k\}, \quad (10)$$

where $TL(i,i) = 0$. Using the algorithm *Total_Loss* whose pseudo–code is given in figure 1, we can compute $TL(0,N)$ in $O(N^3)$ time and $O(N^2)$ space. This algorithm uses tables $TL[0..N, 0..N]$ and $root[0..N, 0..N]$ for storing $TL(i,j)$ and $root(i,j)$ – the root of an optimal binary search tree for $[i,\ldots,j]$, respectively. The pseudo–code for *Optimal_Tree* in figure 1 constructs an optimal tree T_{total} for the interval $[i,j]$ in the additional time $O(j-i)$. ■

```
Total_Loss(N)

BEGIN
for i ← 0 to N do
  for j ← 0 to N do
     TL[i,j] ← ∞
for i ← 0 to N do
   TL[i,i] ← 0
   root[i,i] ← i
for l ← 1 to N do
  for i ← 0 to (N − l) do
    j ← i + l
    for k ← i + 1 to j do
       z ← TL[i,k−1] + TL[k,j]
       z ← z + (j − i + 1)(j + i)/2
       z ← z − k(j − k + 1)
      if z < TL[i,j] then
        TL[i,j] ← z
        root[i,j] ← k
return TL and root.
END
```

```
Optimal_Tree(int i,j,TreeNode *node)

BEGIN
 if i = j then
   value(node) ← i
   left(node) ← NULL
   right(node) ← NULL
   return
 else
   k ← root[i,j]
   value(node) ← k
   value(node) ← k
   left(node) ← new(node)
   Optimal_Tree(i, k − 1, left(node))
   right(node) ← new(node)
   Optimal_Tree(k, j, right(node))
 END
```

Figure 1: Algorithms for finding and explicitly constructing binary search tree minimizing total loss.

Corollary 3.3 *The (perfectly) balanced binary search tree does not minimize the total loss.*

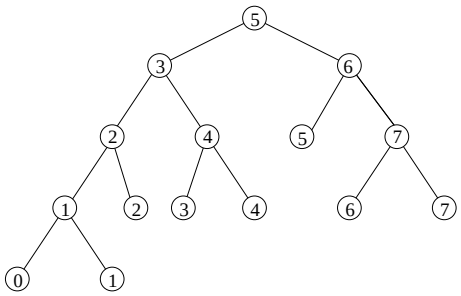


Figure 2: Tree $T_{total}(7)$ that minimizes the total loss on $[0, \ldots, 7]$

Proof: The minimum total loss for $N = 7$,computed by Theorem 3.2, is 33, while the total loss of the balanced binary search on $[0, \ldots, 7]$ is 36.

■

Third, we compute lower and upper bounds for the minimum total loss over the interval $[0, \ldots, N]$.

Lemma 3.4 *Let tl(N) denote the minimum total loss for* $[0, \ldots, N]$. *Then,*

$$\left\lfloor \frac{N}{2} \right\rfloor \left\lceil \frac{N}{2} \right\rceil \leq tl(N) \leq (\lfloor \log N \rfloor + 1) 2^{\lfloor \log N \rfloor} (2^{\lfloor \log N \rfloor} - 1). \tag{11}$$

Proof: From Equation (10), since $tl(N) = TL(0, N)$, we have that

$$tl(N) \geq \min_{1 \leq k \leq N} ((N+1)N/2 - (N+1-k)k) = \left\lfloor \frac{N}{2} \right\rfloor \left\lceil \frac{N}{2} \right\rceil. \tag{12}$$

On the other hand, from Lemma 3.1, and since $tl(x)$ is a non-decreasing function of x and $N \leq 2^{\lfloor \log N \rfloor + 1} - 1$, we have that

$$tl(N) \leq tl(2^{\lfloor \log N \rfloor + 1} - 1) \leq (\lfloor \log N \rfloor + 1) 2^{\lfloor \log N \rfloor} 2^{\lfloor \log N \rfloor} - 1). \tag{13}$$

■

Fourth, we generalize Theorem 3.2 to find an admissible pricing algorithm that minimizes the expected loss, when the maximum buyer's price X has an arbitrary discrete probability distribution on $[0, \ldots, N]$.

Let p be a mass probability function for a random variable on $[0, \ldots, N]$. Then, $p(X)$ is the probability that the buyer's price is equal to X. The probability that the buyer's price

is in the interval $[i, \ldots, j]$, denoted by $p_{i,j}$, is $p_{i,j} = \sum_{X=i}^{j} p(X)$. Let $\overline{X}_{i,j} = \sum_{X=i}^{j} Xp(X)$ – the (unnormalized) conditional expected value of the buyer's price for the interval $[i, \ldots, j]$. The expected loss for a binary search tree T on the interval $[0, \ldots, N]$ with mass probability function p is

$$\overline{loss}(T) = \sum_{X=0}^{N} loss_T(X)p(X). \tag{14}$$

Theorem 3.5 *Given a mass probability function p for the buyer's price in an interval $[0, \ldots, N]$, we can find an admissible algorithm that minimizes the expected loss on this interval, together with the expected loss achieved by that algorithm, in $O(N^3)$ time and using $O(N^2)$ space.*

Proof: Let T be a binary search tree that gives us the minimum expected loss among all binary search trees on $[i, \ldots, j]$. Let k be the root of T, $i+1 \leq k \leq j$. Then,

$$\begin{aligned}
\overline{loss}(T) = &= \frac{1}{p_{i,j}} \sum_{Z=i}^{j} loss_T(Z) \cdot p(Z) = \\
&= \frac{1}{p_{i,j}} \sum_{X=i}^{k-1} loss_T(X) \frac{p(X)}{p_{i,k-1}} p_{i,k-1} + \frac{1}{p_{i,j}} \sum_{Y=k}^{j} loss_T(Y) \frac{p(Y)}{p_{k,j}} p_{k,j} = \\
&= \frac{1}{p_{i,j}} \left(\overline{loss}(T_L)p_{i,k-1} + \overline{loss}(T_R)p_{k,j} - kp_{k,j} + \overline{X}_{i,j} \right),
\end{aligned} \tag{15}$$

where T_L and T_R the left and right subtrees of the root of T respectively. Both T_L and T_R must provide the minimum expected loss on $[i, \ldots, k-1]$ and $[k, \ldots, j]$, respectively. Consequently, we get a recurrence for the minimum expected loss similar to Equation (10).

$$EL(i,j) = \min_{i+1 \leq k \leq j} \{ \frac{1}{p_{i,j}} (p_{i,k-1} EL(i, k-1) + p_{k,j} \cdot EL(k,j) - k \cdot p_{k,j} + \overline{X}_{i,j}) \}, \tag{16}$$

with $EL(i,i) = 0$, where $EL(i,j)$ denotes the minimum expected loss on $[i, \ldots, j]$. This recurrence can be solved for $EL(0,N)$ in $O(N^3)$ time and $O(N^2)$ space.

■

In appendix A, we show sample trees that minimize the expected loss for various probability distributions, and in appendix B we show the total loss and height of optimal trees as a function of N.

4 Bounding the Maximum Loss

We consider the problem of minimizing the maximum loss. Dynamic programming is not applicable to the problem of computing the minimum maximum loss, since this problem does not have the optimal substructure property.

First, we compute lower and upper bounds for the minimum maximum loss as a function of N.

Lemma 4.1 *Let $ml(N)$ denote the minimum maximum loss on $[0,\ldots,N]$. Then,*

$$N-1 \leq ml(N) \leq \lfloor \log N \rfloor (2^{\lfloor \log N \rfloor + 1} - 2). \tag{17}$$

Proof: Recall that each admissible pricing algorithm is able to recognize every number in $[0,\ldots,N]$. Thus, such an algorithm has to distinguish between $N-1$ and N. The only question that can achieve this is: "Is $N \leq X$?". The answer to this question is "Yes", if $X = N$, and is "No", if $X = N-1$. Thus, in any binary search tree T, $loss_T(N-1) \geq N-1$.

Let $k = \lfloor \log(N) \rfloor + 1$. Consider the (perfectly) balanced binary search tree T on $[0,\ldots,2^k-1]$. Let $k = \lfloor \log(N) \rfloor + 1$. We compute an upper bound for the maximum loss on T over the interval $[0,\ldots,2^k-1]$ as follows:

$$max\_loss_T = \max_{0 \leq X \leq 2^k-1} loss_T(X) = \max_{0 \leq X \leq 2^k-1} \sum_{i=0}^{k-1} loss_i(X) \leq k(2^k-1), \tag{18}$$

where $loss_i(X)$ denotes the loss of the search for X on the i-th step in T. Observe that for every X, that the maximum loss at the i-th step does not exceed $N-1 = 2^k-2$, and that there is a step at which no loss occurs. Hence, since $loss_{\mathcal{A}}(0) = 0$ for any pricing algorithm $\mathcal{A}$, we have that $max\_loss_T \leq (k-1)(2^k-2)$.

■

Note that the maximum loss on a given interval, as the worst case measure, is the same for any probability distribution function for the buyer's valuation on that interval.

Furthermore, the (perfectly) balanced binary search tree does not minimize the maximum loss. For example, the tree $T_{total}(15)$, that minimizes the total loss on the interval $[0,\ldots,15]$, has maximum loss equal to 25. However, the (perfectly) balanced binary search tree T has maximum loss equal to 28.

Theorem 4.2 *There is no strategy that minimizes both the total loss and the maximum loss functions at the same time.*

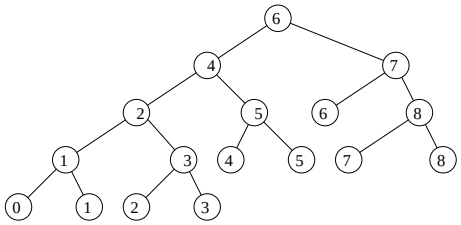


Figure 3: Tree $T_{\max}(8)$ that minimizes the maximum loss on $[0, \ldots, 8]$ and has the minimum total loss among all those trees with the same maximum loss.

Proof: By performing an exhaustive search among all such binary search trees, as stipulated in Lemma 2.1, for each N, $1 \leq N \leq 8$, we found a tree that minimizes the total loss among all the trees that minimize the maximum loss. For $N = 8$ the minimum maximum loss is equal to 8, see figure 3; the minimum total loss among the trees with maximum loss 8 is 47. On the other hand, the minimum total loss on $[0, \ldots, 8]$, computed by the algorithm in figure 1 is 46. Therefore, minimizing the total loss does not necessarily minimize the total loss and vice versa.

■

5 Maximizing the Total Profit for Limited Sales Period and Limited Supply

In this section we consider our initial model with additional constraints, and also relax the constraint that the buyer's valuation is constant over time. We consider constraints that limit the number of units available for sale *(supply)* and/or the time to conduct the sales *(sales period)*. These constraints make the maximum possible profit finite. Therefore, the loss functions, as they were defined in section 2, are not appropriate indicators of the "goodness" of a pricing algorithm. However, in many cases it is still possible to create pricing algorithms that explicitly maximize the average profit. We can also find optimal pricing algorithms, when the buyer's valuation is a deterministic function (instead of being constant) for a fairly broad class of valuation functions.

A typical situation with limited sales period and supply, and variable buyer's valuation follows. The seller has M units of a product for sale, that he bought at a price of C dollars per unit from a manufacturer. To sell them the seller rents a shop and warehouse space, which he can use for T months, but he can quit after any number of months less than T. If the seller decides to go out of business he always can return the unsold product to the manufacturer for the full refund. In the beginning of every month the seller pays a flat rent for the shop, and the warehouse rent which is proportional to the number of remaining units of the product. The sales are conducted under the same protocol as before with the time

step equal to a month: at each time step the seller posts the current price per unit of the product. The buyer has the secret maximum price he is willing to pay and the deal will occur if and only if the seller's price does not exceed the buyer's price. The buyer does not change the secret price during the sales period.

Although the sales period and supply bounds are explicit in the example above, it may be not clear why the buyer's valuation is variable. Recall that "the buyer's valuation" actually means the difference between the maximum price the buyer is willing to pay and the current seller's cost per unit of the product. Hence, changes in seller's cost can affect the buyer's valuation.

The main question is: What is the best pricing strategy for the seller under the conditions above, if he knows that the secret maximum buyer's price is somewhere between C and $C + N$ dollars per unit?

The answer to this question can be obtained as a specific case of Theorem 5.1, the main result of this section. The corollaries of this theorem and the simple algorithm from 5.3 enable us to construct optimal pricing algorithms for a wide variety of market models. Namely, we present a dynamic programming construction of a weakly admissible algorithm that maximizes the total profit for limited sales period. We also show a simple algorithm that maximizes the total profit when the sales period is unlimited or sufficiently long, the initial supply of the seller's product is limited, and the buyer's valuation is constant.

First, we introduce necessary notations and definitions that we use throughout this section. We assume that, for a given sales period of length T and a given initial interval of buyer's valuations $[0, \ldots, N]$, there exists a family of *valuation change functions* $\mathcal{G} = \{g_0, g_1, \ldots, g_{T-1}\}$ such that

$$X_0 \in [0, \ldots, N] \text{ and } X_{t+1} = g_t(X_t, m_t), \tag{19}$$

and this family of functions is known to the seller in advance.

The actual profit made by a pricing algorithm $\mathcal{A}$ is given by

$$\mathit{profit}_{\mathcal{A}}(X_t, m_t) = \begin{cases} p_t, & \text{if } p_t \leq X_t \text{ and } m_t > 0 \\ 0 & \text{otherwise,} \end{cases} \tag{20}$$

where m_t is the number of available units of the product for sale at time t (m_t could be equal to ∞ for the case of unlimited supply). Furthermore, if the number of supply units is limited, then $M_t = \begin{cases} m_t = m_{t-1} - 1, & \text{if } m_{t-1} > 0 \text{ and } p_{t-1} \leq X_t \\ m_{t-1}, & \text{otherwise} \end{cases}$

The profit of a pricing algorithm over a sales period $[t_1 \dots t_2]$ is defined as

$$profit_{\mathcal{A}}(X_{t_1}, t_1, t_2) = \sum_{t=t_1}^{t_2} profit_{\mathcal{A}}(X_t, m_t).$$

The *total profit* of a pricing algorithm with sales period $[t_1, \dots, t_2]$ of length $t_2 - t_1 + 1$, when the range of possible buyer's valuations at time t_1 is $[i, \dots, j]$ is defined by

$$total\_profit_{\mathcal{A}}(i, j, t_1, t_2, m_{t_1}) = \sum_{X=i}^{j} profit_{\mathcal{A}}(X, t_1, t_2) \tag{21}$$

Figure 4: Non-admissible tree $T(2, 3, 1)$ (left), that maximizes the total profit achievable on $[2, 3]$ when the sales period has length 1, *vs* an admissible but suboptimal tree (right).

Note that a pricing algorithm can still be represented as a binary search tree, although an optimal pricing algorithm may be not admissible (see figure 4).

We call a pricing algorithm $\mathcal{A}$ *weakly admissible* if, for every $t \geq 0$,

1. $P_t \in [i + 1, \dots, j]$, or
2. $P_t = i$, or
3. $P_t = j + 1$,

where $\mathcal{P}_t = [i, \dots, j]$. Thus, a weekly admissible algorithm either makes progress towards finding the true buyer's valuation as an admissible algorithm does (case 1), or it makes a deal for sure (case 2) with the maximum guaranteed profit, or it "deliberately" refuses to make a deal at a given time (case 3).

Note that for every pricing strategy there exists a corresponding weakly admissible pricing algorithm that provides at least the same profit. Thus, if the maximum profit is finite, then there exists an optimal weakly admissible algorithm. Also note that when the sales period is finite, the height of the binary search tree that corresponds to a weakly admissible algorithm is equal to the length of the sales period.

Theorem 5.1 ***(Bounded sales period and bounded supply.)*** *Let T be the length of the sales period, let M be the initial supply of product units, and let $[0, \ldots, N]$ be the interval of the initial buyer's valuation. Given the family of valuation change functions $\mathcal{G} = \{g_0, g_1, \ldots, g_{T-1}\}$, such that each one of them is non-decreasing in X, we can construct a weakly admissible algorithm that maximizes the total profit.*

Proof: Let T_{total} be the binary search tree corresponding to a weakly admissible pricing algorithm that maximizes the total profit. Suppose that at time t, X_t is in $[i, \ldots, j]$. Let $tp(i, j, t, T-1, m_t)$ denote $total\_profit_{T_{total}}(i, j, t, T-1, m_t)$. Let k be the root of T_{total}, $i \leq k \leq j+1$. Let T_L and T_R be the left and the right subtree of the root, respectively.

In order to compute the total profit obtained at time t, we consider two cases:

Case 1: There is a deal at time t. Then $X_t \geq k$ and $profit(X_t, m_t) = k$. Consequently, $m_{t+1} = m_t - 1$ and $g_t(k, m_t) \leq X_{t+1} \leq g_t(j, m_t)$, because g_t is a non-decreasing function in X. Note that if $k = i$, then a deal always occurs.

Case 2: There is no deal at time t. Then $profit(X_t, m_t) = 0$, $m_{t+1} = m_t$ and $g_t(i, m_t) \leq X_{t+1} \leq g_t(k-1, m_t)$. Note that if $k = j+1$, then no deal occurs.

The total profit obtained at time t is given by

$$\sum_{X_T=i}^{j} profit(X_t, m_t) = \sum_{X_T=i}^{k-1} profit(X_t, m_t) + \sum_{X_T=k}^{j} profit(X_t, m_t) = (j-k+1)k.$$

Next, we prove the optimal substructure property for weakly admissible pricing algorithms that maximize the total profit. We show that T_L has to provide the maximum total profit for $X_{t+1} \in [g_t(i), \ldots, g_t(k-1)]$ and sales period $[t+1, \ldots, T-1]$, when there are m_t remaining units for sale. Assume the contrary, *i.e.* let some other binary search tree T_L^* provide the greater profit than T_L does under the conditions above. Then, we replace T_L with T_L^* as the left subtree in T_{total} and we get greater profit than $tp(i, j, t, T-1, m_t)$. Contradiction. Analogously, T_R has to provide $tp(g_t(k, m_t), g_t(j, m_t), t+1, T-1, m_t - 1)$.

A dynamic programming algorithm that constructs an optimal weakly admissible pricing algorithm that maximizes the total profit follows from the recurrence:

$$tp(i, j, t, T, m_t) = \begin{cases} 0, \text{ if } t = T \text{ (the sales period has ended, no deal is possible)} \\ 0, \text{ if } m_t = 0 \text{ (there is no more product to sell)} \\ 0, \text{ if } i > j \text{ (the respective tree is empty)} \\ i, \text{ if } i = j,\ t < T \text{ and } m_t > 0 \text{ (the algorithm knows that } X_t = i) \\ \max\limits_{i \leq k \leq j+1} \{tp(g_t(i, m_t), g_t(k-1, m_t), t+1, T-1, m_t) + \\ \quad tp(g_t(k, m_t), g_t(j, m_t), t+1, T-1, m_t - 1) + (j-k+1)k\}, \text{ otherwise.} \end{cases} \quad (22)$$

This recurrence can be solved in $O(\bar{N}^3TM)$ time and $O(\bar{N}^2TM)$ space, where $\bar{N}$ is the maximum possible buyer's valuation for $t = 0, 1, \ldots, T$ and any possible initial value X_0. Note that $\bar{N}$ depends N and $\mathcal{G}$.

■

Corollary 5.2 ***(Bounded sales period and unbounded supply.)*** *If the number of available units for sale is unlimited and the sales period has finite length T, then we can construct an weakly admissible algorithm that maximizes the total profit.*

Proof: Since the seller is not allowed to sell more than one unit of the product at each unit of time, the length of the sales period is an upper bound on the number of units which can be sold. The corollary follows from Theorem 5.1 with $M = T$.

■

Remark 1: If the maximum possible buyer's valuation is bounded by a polynomial of N for every moment of time t within the sales period and any possible initial buyer's valuation, then the time (and space) to construct the algorithm in Theorem 5.1 is polynomial in N, M and T. In particular, when buyer's valuations do not increase in time, such a construction requires $O(N^3TM)$ time and $O(N^2TM)$ space. Since the accumulated seller's costs and some other factors (e.g., the loss of novelty of the product), we expect that in practical applications the buyer's valuation will be non-increasing in time.

Theorem 5.3 ***(Bounded supply, unbounded sales period, and constant buyer's valuation.)*** *Let the buyer's valuation be a constant in $[0, \ldots, N]$. If there are M units of a product for sale and the sales period is unlimited, then the seller can obtain the maximum possible profit $M \cdot X$ in $O(N + M)$ time.*

Proof: The seller can get the maximum possible profit $M \cdot X$, if he sets the initial price $P_0 = N$ and

$$P_t = \begin{cases} P_{t-1} - 1, & \text{if } P_{t-1} > X \\ P_{t-1}, & \text{if } P_{t-1} \leq X \end{cases}$$

■

Note that if the sales period is long enough ($T > N + M$), Theorem 5.3 provides a pricing algorithm that is simpler than the one from Theorem 5.1, and it also maximizes the total profit for bounded supply, bounded time, and constant buyer's valuations.

Remark 2: (Implicit time bounds) If the seller has unlimited supply and unlimited sales period, but the buyer's valuation function is diminishing[5] for any possible initial value, then we still can use Theorem 5.1 to get the optimal algorithm, because we can obtain a bound on effective length of the sales period.

Finally, for the case where the buyer's utilities increase in time, currently, we do not have any pricing algorithms for either bounded or unbounded supply of a product. Since both the profit and the loss may grow indefinitely, an optimal pricing algorithm may defer a deal for a non–constant amount of time. Fortunately, this scenario seems to be unlikely in practice.

6 Conclusions

We introduce a simple market model for buyer/seller sales transactions that captures the essence of electronic commerce over the Internet, and consider the problem of pricing algorithms for the seller that maximize his/her profits. Using the idea of loss functions, we provide a polynomial–time algorithm that finds a pricing strategy for the seller that minimizes the cumulative (total) loss (lost profit because no sale occurred), when the buyer's valuation is an unknown constant in a given range and the sales period is unbounded. We also compute lower and upper bounds on the total loss. Preliminary results for the problem of minimizing the maximum loss incurred at any time are also presented.

We also present algorithms for finding pricing strategies that minimize the total loss when there is a limited sales period, or a limited supply of a product for sale, or the buyer's valuation is a non–increasing function of time (known to the seller).

In future work , we plan to extend the simple market model and pricing strategies presented here to also consider multiple buyers with valuations that vary over time. We also plan to design algorithms for finding pricing strategies that minimize the maximum loss for the market model in this paper.

References

[1] L.M. Ausubel and R.J. Deneckere. Reputation in bargaining and durable goods monopoly. *Econometrica*, 57(3):511–531, 1989.

[5] We say that the non-negative function is *diminishing*, if it is decreasing and its limit value is 0. For a discrete function f, it means that f becomes 0 after a finite amount of time.

[2] Baruch Awerbuch and Yossi Azar. Blindly-competitive algorithms: Pricing and bidding a case study. Extended Abstract, April 1995.

[3] Shai Ben-David, Eyal Kushilevitz, and Yishay Mansour. Online learning versus offline learning. In P. Vitanyi, editor, *Proceedings of the second EuroCOLT*, pages 38–52, 1995.

[4] R. Benabou. Optimal price dynamics and speculations with a storable good. *Econometrica*, 57:41–80, 1989.

[5] Avrim Blum. On-line algorithms in machine learning. Technical report, Carnegie Mellon University, 1996.

[6] N. Cesa-Bianchi, Y. Freund, D. P. Helmbold, and M. Warmuth. On-line prediction and conversion strategies. In John Shawe-Taylor and Martin Anthony, editors, *Proceedings of EuroCOLT'93*, pages 205–216, 1993.

[7] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to algorithms.* The MIT Press, 1993.

[8] R. El-Yaniv, A. Fiat, R. Karp, and G. Turpin. Competitive analysis of financial games. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 327–333, 1992.

[9] M. Harris and A. Raviv. A theory of monopoly pricing schemes with demand uncertainty. *American Economic Review*, 71:347–365, 1981.

[10] Jyrki Kivinen and Manfred K. Warmuth. Using experts for predicting continuous outcomes. In John Shawe-Taylor and Martin Anthony, editors, *Proceedings of EuroCOLT'93*, pages 109–120, 1993.

[11] R.A. Meyer. Monopoly pricing and capacity choice under uncertainty. *American Economic Review*, 65:326–337, 1975.

[12] J. Sobel. Durable goods monopoly with with entry of new consumers. *Econometrica*, 59(5):1455–1485, 1991.

[13] V. Vapnic. Structure of statistical learning theory. In Alex Gammerman, editor, *Computational Learning and Probabilistic Reasoning*, pages 3–31. John Wiley & Sons Ltd., 1996.

A Sample Trees

We show binary search trees that minimize the expected loss, together with the expected loss achieved, when the buyer's valuation X has the uniform p.m.f. $U(a, b)$, or X is the discretization (via rounding) of a continious random variable with normal p.d.f. $N(\mu, \sigma)$. From Fig. 5, we observe that **(i)** The expected loss for more skewed p.d.f's is less, but the height of the corresponding tree is larger, since we can afford to have less probable X at larger depths (incurring greater $loss(X)$). **(ii)** The expected loss is an increasing function of the expected value of X.

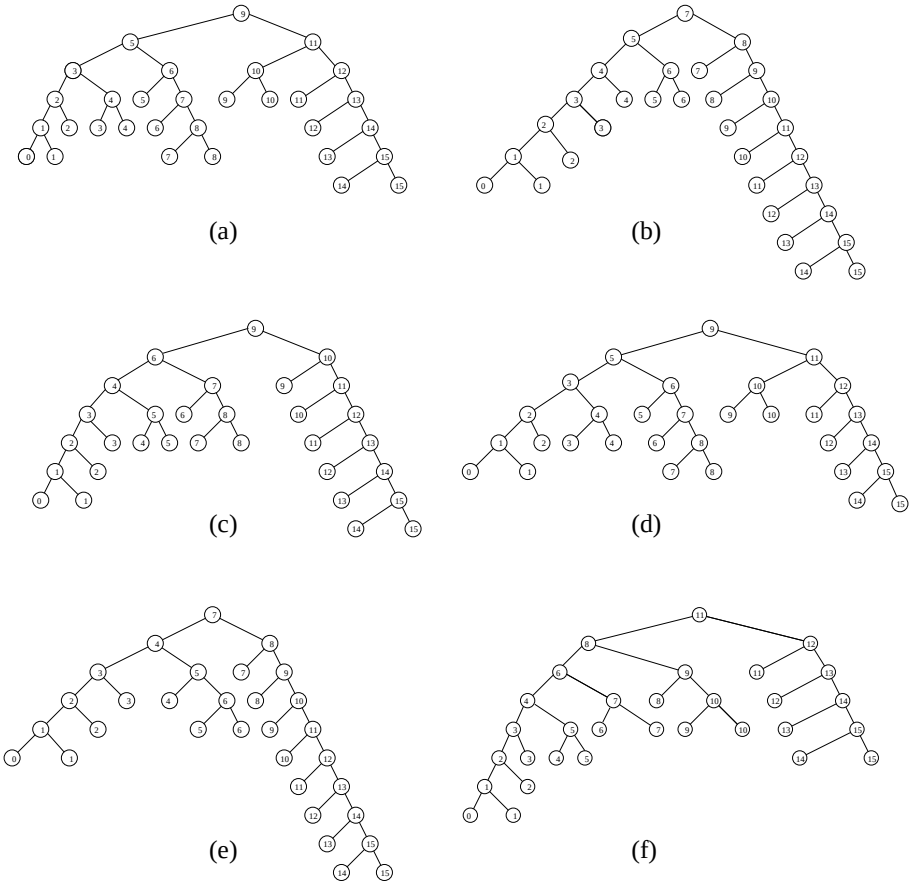
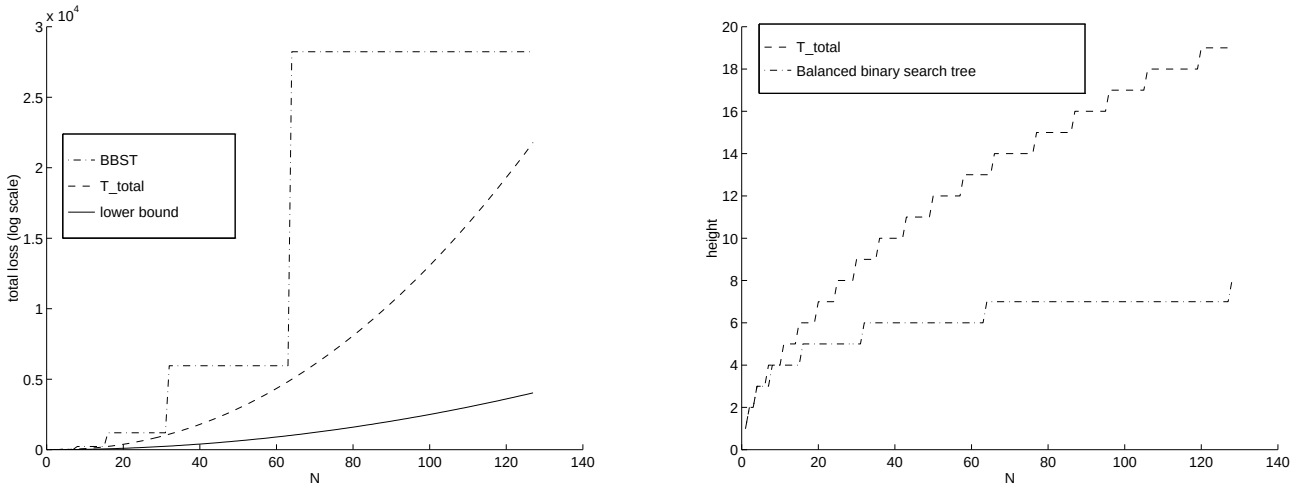


Figure 5: (a) $X \sim U(0, 15)$; $exp\_loss = 12.5625$; (b) $X \sim N(7.5, 2)$; $exp\_loss = 10.436841$; (c) $X \sim N(7.5, 4)$; $exp\_loss = 12.563369$; (d) $X \sim N(7.5, 8)$; $exp\_loss = 12.734426$; (e) $X \sim N(4, 4)$; $exp\_loss = 8.7110581$; (f) $X \sim N(12, 4)$; $exp\_loss = 14.764054$.

B Total Loss and Tree Height



(a) (b)

Figure 6: **(a)** Minimum total loss on $[0, \ldots, N]$ as a function of N.(BBST: total loss achieved by the (perfectly) balanced binary search tree on $[0, \ldots, N]$.) The lower bound is computed based on Lemma 4.1. **(b)** Height of the tree minimizing the total loss $[0, \ldots, N]$ computed by the algorithms in figure 1, as a function of N.