Finite Optimal Stopping Problems: The Seller's Perspective

Mehdi Hemmati[1] *and J. Cole Smith*[1]

Invited

Abstract:

We consider a version of an optimal stopping problem, in which a customer is presented with a finite set of items, one by one. The customer is aware of the number of items in the finite set and the minimum and maximum possible value of each item, and must purchase exactly one item. When an item is presented to the customer, she or he observes its value, and determines whether to purchase the item or to permanently dismiss the item. The customer's objective is to maximize the value of the purchased item. In this paper, we consider the problem from the perspective of the seller, who wishes to maximize profit associated with the sold item. Hence, the seller seeks an optimal sequence of items to sell, given that the customer acts according to some near-optimal decision-making rules. Our paper takes the perspective that the customer may not act optimally due to imperfect decision-making strategies and/or to the seller's uncertainty in the items' values to the customer. We consider max-min and max-expectation objectives when customer behavior is not completely predictable, and discuss the problem tractability in these cases.

Keywords:

optimal stopping, dynamic programming, optimization, mathematical programming

[1]University of Florida

http://dx.doi.org/10.7771/1932-6246.1092

1. Introduction and Problem Overview

We begin this paper by describing the following stopping problem, which is a classical problem that has received substantial attention across several disciplines. A customer must purchase one item out of a set of items that are presented one at a time to the customer. When an item is presented to a customer, the customer evaluates its value, and must decide whether to purchase the item (thus ending the game) or permanently discard (or "reject") the item. The customer's objective is to maximize the value of the purchased item. (The reward is equal to the value of the purchased item, and is independent of all other items' values.) The customer is aware at the beginning of the game of the number of items that will be presented (*n*) and the probability distribution used to generate the items' values. Observe that if one item remains, the customer must purchase it. It is thus straightforward to see that this model captures the case in which the customer does not need to buy an item, which we would allow by simply letting the final item have a value equal to the customer's value of purchasing nothing at all.

This game is a special case of the broad class of *optimal stopping problems* in which the customer must determine when to stop the game (e.g., by purchasing an item). See Chow, Robins, and Siegmund (1971) for an early summary of optimal stopping problem analysis, and Ferguson (2010) for a comprehensive modern study of this class of problems. In fact, the original "secretary problem" (see, e.g., Ferguson, 1989; Freeman, 1983; Gardner, 1960; Samuels, 1991) is given as above, but where the objective is to pick the most-valuable item from among the set of all items. There is no reward for picking any other item than the best one. There are numerous versions of these games, including variations in which *n* is unknown or infinite, in which rewards are given for solutions other than the most-valuable one, and in which an attempt to purchase the item may fail (Bartoszynski & Govindarajulu, 1978; Gilbert & Mosteller, 1966; Pressman & Sonin, 1972; Smith, 1975).

The stopping problem considered in this paper satisfies the so-called memoryless property, in the sense that prior actions that occurred in this game do not affect the remainder of the game. The customer only needs to know how many items remain and the value of the next item (in addition to the boundary conditions of the game) to make the next decision; prior information regarding the values of previous (rejected) items is irrelevant. This memoryless property enables us to employ dynamic programming techniques (see, e.g., Bellman, 1957) to solve the above stopping problem. (We provide the technical details for this algorithm in Section 2.)

In this paper, we introduce a new problem from the seller's perspective. In this problem, each item is also associated with a *profit* (independent from the item's value to the customer) that the seller makes if the customer purchases the item. The seller must determine a sequence of the items to present to the customer, so that the customer (acting rationally, i.e., optimally in his or her own best interests) would choose an item that results in a maximum possible profit to the seller. This problem is nontrivial, because placing the most-profitable items too early in the sequence may result in the customer rejecting those items, in hopes of finding an item with more value to the customer later in the sequence. Placing the most-profitable items late in the sequence incurs the risk of having the customer terminate search before encountering these items.

Moreover, this problem is further compounded by the fact that human decision makers tend *not* to optimally solve stopping problems in laboratory settings. We refer the reader to an excellent introductory treatment of this material in Beardon and Rapoport (2005), and to recent results appearing in Bearden, Rapoport, and Murphy (2006), Lee (2006), and Seale and Rapoport (1997). A common theme in this line of literature is that decision makers tend to terminate their search too soon. It is potentially very risky for the seller to assume that a customer will act rationally, in the sense that the customer follows an exact optimization algorithm in selecting an item.

We present an example to illustrate the situation. Suppose that $n = 6$, and that the customer believes that the items' values are uniformly distributed in the interval [0, 100]. Items 1, . . . , 6 are arranged in non-increasing order of profit to the seller (so that item 1 is most preferred). Consider the situation given in Figure 1, which depicts a sequence that the seller has chosen to present to a customer. The seller has evidently gambled that the customer will reject item 5, which has a value of 70 to the customer. Ideally, the customer would then purchase item 1, resulting in the most profit to the seller.

Indeed, an optimal customer will reject item 5 with six items remaining, and purchase item 1 with five items remaining. (In Section 2, we will illustrate an optimal decision-making framework for the customer. In particular, it turns out that an optimally behaving customer would buy the sixth-to-last item if its value exceeds 77.5, and would buy the fifth-to-last item if its value exceeds 74.2.) However, the seller risks having a conservative decision maker (rather than an optimal customer) who stops the process too early and purchases item 5, or stops the process too late and purchases item 6.

The challenge in this paper is to analyze the seller's problem from three different perspectives. We begin in Section 2 by assuming that the customer acts in an optimal manner, breaking ties in a manner that is disadvantageous to the seller. That is, when the customer's decision is optimal to either reject or purchase an item, the customer takes the opposite action desired by the seller. (This pessimistic assumption can easily be modified.)

We then accommodate the stochastic nature of human decision makers by accounting for randomness in the customer's optimal decision-making policies, and in the customer's perception of the items' values. For instance, we can model a decision maker who tends to stop too soon by adjusting the true item values to be higher than they actually are. However, when dealing with uncertainty, we must specify an objective that reflects the seller's optimization philosophy. A risk-averse seller may attempt to maximize the minimum profit that can be made, given a boundedly rational customer. We discuss this problem, along with a simple model for bounding customer rationality, in Section 3.

Figure 1. Sequence of items in an optimal stopping problem.

Item number	Customer value
5	70
1	75
6	65
2	80
3	30
4	20

Order presented to the customer ↓

Although a seller who is playing this game once may prefer an outcome with limited risk of selling a low-profit item, a seller who plays this game repeatedly would more likely prefer to maximize *expected* profit instead. Hence, in Section 4, we present a problem in which the seller maximizes expected profit given a discrete probability distribution function of the customer's problem-solving parameters. We demonstrate that even restricted versions of the expected-value problem are NP-hard, and are thus quite difficult to solve in the worst case. Finally, we conclude in Section 5 and discuss avenues for future research.

2. Seller's Problem with an Optimal Customer

We begin by introducing notation for this problem and presenting the optimal dynamic programming strategy for the customer. For each item $i = 1, \ldots, n$, the item's value to the customer is v_i, and the item's profit to the seller is b_i. For the sake of simplicity, we examine the case in which the customer assumes that item values are generated from the uniform distribution on the interval [0, 100]. The lower and upper bounds given here are arbitrary, and the following discussion easily accommodates any generic (finite) bounds. Also, the logic behind our procedures does not change if the values are assumed to be non-uniformly generated, so long as conditional expectations are finite.

As stated in Section 1, we assume that the customer's decisions are not affected by the values of the previously seen items, but only by the number of remaining items. More precisely, the customer's dynamic programming algorithm employs backward recursion, starting from the situation in which only one item remains. In this case, the customer must purchase the item, and the customer's expected value will be 50. Now, if there are 2 items remaining in the set, the customer will purchase the item if its expected value is at least 50, and will reject the item otherwise. When 2 items remain, there is a 50% chance that the customer rejects the item, because its value does not exceed 50 (leaving the customer with an expected item value of 50 from the last item), and a 50% chance that the customer accepts the item because its value belongs to the interval [50, 100] (yielding an expected value of 75 from the second item). The customer's overall expected value with 2 items left is thus 62.5.

In general, when examining the *i*th item in the sequence, we define t_i to be the customer's expected value from purchasing an item. A rational (or *optimal*) customer applies the following recursive formula to compute these values:

$$t_n = 50, \tag{1a}$$

$$t_i = \Pr(V > t_{i+1})\left(\frac{t_{i+1} + 100}{2}\right) + \Pr(\mathrm{V} \le \mathrm{t}_{\mathrm{i+1}})t_{i+1} \quad \forall i = 1, \dots, n-1, \tag{1b}$$

where V is the uniform random variable reflecting the value of any item; hence, $\Pr(V \le k)$ is 1 if $k \ge 100$, 0 if $k \le 0$, and $k/100$ otherwise, with $\Pr(V > k) = 1 - \Pr(\mathrm{V} \le k)$. (Note that this process can be adapted for non-uniform distributions and for distributions that depend on how many items remain.) Hereafter we refer to *t*-values as *thresholds*, because they reflect how an optimal customer makes decisions: if the *i*th item's value offered to the customer exceeds t_{i+1}, then the customer selects it. For notational convenience (allowing us to handle the case corresponding to the last item), we define $t_{n+1} = -\infty$.

Without loss of generality, we assume $b_1 \ge \cdots \ge b_n$. We discuss the seller's problem in terms of assigning items to slots in the sequence presented to the customer. The seller's goal will be to induce the customer to buy item 1 if possible, and failing that, to buy item 2, and so on. Conceptually, the seller will place item 1 in the earliest slot *i* such that $v_1 > t_{i+1}$, and item 1 is purchased if the customer reaches slot *i* in the search. Accordingly, we define

$$p_i = \min\left\{p \in \{1, \dots, n\} : v_i > t_{p+1}\right\} \quad \forall i = 1, \dots, n. \tag{2}$$

For item $i = 1, \dots, n$, p_i represents the earliest possible slot in any sequence in which the customer would choose item *i*, assuming that the search is still active. Of course, if $p_1 = 1$, the problem is trivial: place item 1 in the first slot, with the remaining slots arbitrarily determined, and the customer selects item 1. Otherwise, the seller seeks to place lessprofitable items early in the sequence whose values are small enough that the customer rejects them, until reaching item 1. Define:

$$N_p = \left\{ i \in \{1, \ldots, n\} : v_i < t_{p+1} \right\} \quad \forall p = 1, \ldots, n. \tag{3}$$

The set N_p contains the items that would not be chosen by the customer if they are positioned in slot p. The following proposition is useful in devising our solution method:

Proposition 1. Suppose that no solution exists in which an item in $\{1, \ldots, i-1\}$ can be sold to the customer, for some $1 \le i < n$, and define $\Gamma = \min\{p_i - 1, i-1\}$. Then item i can be sold to the customer if and only if a sequence exists in which item i is placed at slot p_i, and items $j = 1, \ldots, \Gamma$ are placed in any arbitrary order in slots $p_i - \Gamma, \ldots, p_i - 1$. Also, if $p_i > i$, then an item in N_p is placed at slot p, for all $p = 1, \ldots, p_i - i$, with no item appearing twice in slots $1, \ldots, p_i$.

Proof. First, suppose that the conditions of this proposition hold true. The customer will purchase item i positioned at p_i if no item is selected before p_i. By assumption the customer rejects any item $h < i$. The only items $j > i$ placed in a slot $p < p_i$ obey the condition $j \in N_p$, and so the customer will reject these items as well, thus purchasing item i in slot p_i.

Now, suppose that it is possible to sell item i (but not items in $\{1, \ldots, i-1\}$). We first establish that a schedule exists in which item i is placed in position p_i. If i is placed before p_i, the customer will not purchase it. Else, suppose that a schedule exists in which item i is placed in slot $k > p_i$. Then by switching the items in slots k and p_i, the customer would still purchase item i in p_i. Given that item i is placed in slot p_i, we next establish that a schedule must exist in which each item $j \in J = \{1, \ldots, \Gamma\}$ is placed in slots $p_i - \Gamma, \ldots, p_i - 1$ (in any order). Because items in J cannot be purchased by the customer, they can be placed anywhere in the sequence. Suppose that an item $j \in J$ is not one of the $|J|$ items that immediately precedes item i in position p_i, and swap j with an item $k \notin J$ that is currently positioned in some slot $p \in \{p_i - \Gamma, \ldots, p_i - 1\}$. If j had followed item i, then the customer would reject j instead of k in slot p. Otherwise, $p \in \{1, \ldots, p_i - i\}$, and item k is moved earlier in the sequence. If item k was rejected in its original position, it would also be rejected at an earlier position. The customer still rejects item j (in any position), and because the remaining items are unaffected, would still purchase item i. After executing all such swaps, and rearranging items in J as necessary, we obtain a schedule satisfying the conditions of the proposition. □

Using Proposition 1, the seller can employ the following greedy principle. Starting with item 1, examine whether there exists a sequence in which item 1 is placed in p_1 and the earlier positions $p = 1, \ldots, p_1 - 1$ contain items belonging to N_p. If so, an optimal solution can be constructed based on this sequence. Otherwise, the seller attempts to sell item 2, and so on, until an item can be sold.

Computationally, our challenge is to solve the problem of placing items in slots $1, \ldots, p_i - i$ (when $p_i > i$) as efficiently as possible. First, it is useful to define a maximum "safe" slot for each item i as:

$$f_i = \begin{cases} 0 & \text{if } v_i \geq t_2 \\ \max\left\{p \in \{1, \ldots, n\} : v_i < t_{p+1}\right\} & \text{otherwise.} \end{cases} \quad \forall i = 1, \ldots, n, \qquad (4)$$

which indicates that item i cannot be purchased in slots $1, \ldots, f_i$, where $f_i = 0$ means that item i would be purchased in any slot. (Note that $f_i = p_i - 1$ if $v_i \neq t_p$ for all $p = 2, \ldots, n$.) Now, suppose that we sort the f-values into a list F in non-decreasing order of their values, discarding those i for which $f_i = 0$. We now state the following lemma, related to ordering the first items of a sequence.

Lemma 1. Suppose that no solution exists to a given problem in which any item in the set $\{1, \ldots, i-1\}$ can be purchased by the customer. Further suppose that $p_i > i$, and that there exists an (optimal) sequence with item i placed in slot p_i, which induces the customer to purchase item i. Then there exists a solution in which items scheduled in positions $1, \ldots, p_i - i$ appear in the same order as they appear in F.

Proof. Consider a sequence with i positioned in p_i, and a subset of items in the set $\{i + 1, \ldots, n\}$ in positions $1, \ldots, p_i - i$ (such a solution is known to exist by Proposition 1). Items in the first $p_i - i$ slots having the same f-value can clearly be rearranged to meet the sequence of items in F. Now, consider items j and k with $f_j > f_k$, but item j (in slot p') ordered before item k (in slot p''). Suppose that we swap these items' slots. Item k in slot p'' would still be rejected, because item k was rejected in the later slot of p''. Item j would be rejected in slot p'' because k was rejected in the same slot, and the fact that $f_j > f_k$ indicates that $v_j < v_k$. With a lesser value to the customer, the customer would also reject j in slot p''. □

We can now execute the following algorithm to optimize the seller's problem, which considers only sequences of the form given in Proposition 1, whose first $p_i - i$ slots (if $p_i > i$) are scheduled according to Lemma 1. We formally state this process in Algorithm 1.

The first while-loop in Algorithm 1 establishes list, which aids us in maximizing the number of items that are to be placed at the front of the sequence. After this first step is complete, the second while-loop terminates when $j + |\text{list}|$ is at least as large as p_j. When this happens, scheduling a combination of items on list (in positions $p < p_j$) and items

Algorithm 1—Seller's Problem with an Optimal Customer

```
1: Compute $p_i$ and $f_i$, $\forall i = 1, \ldots, n$. Obtain $F$ by sorting the items in non-decreasing
   order of their values, excluding item 1, and breaking ties in decreasing order of
   item index.
2: Initialize $j = 1$ and list $= \emptyset$; let $|\text{list}|$ denote the number of elements in list.
3: **while** $j < n$ **do**
4:     Pick the next item of $F$, and call it item $k$.
5:     **if** $f_k \geq |\text{list}| + 1$ **then**
6:         Add $k$ to the end of list.
7:         Remove item $k$ from $F$.
8:     **end if**
9:     Set $j = j + 1$.
10: **end while**
11: Set $j = 1$.
12: **while** $p_j > j + |\text{list}|$ **do**
13:     Set $j = j + 1$.
14:     **if** item $j$ belongs to list **then**
15:         Let $pos_j$ be the position of item $j$ in list.
16:         Delete item $j$ from list.
17:         **if** there exists an item $k \in F$ with $f_k \geq pos_j$ **then**
18:             Find an item $k$ having the smallest value of $f_k$ in $F$, subject to $f_k \geq pos_j$.
19:             Insert item $k$ into position $pos_j$ of list, and delete $k$ from $F$.
20:         **end if**
21:     **end if**
22: **end while**
23: An optimal sequence schedules items as in Proposition 1, with the ordering of
    the first $\min\{p_j - 1, j - 1\}$ items given by list.
```

$1, \ldots, j$ results in a sequence in which the customer buys item j. If this condition is not satisfied, then we instead attempt to sell item $j + 1$. But first, if item $j + 1$ had been a part of list, which preceded j at the previous iteration, we must remove it from its tentatively scheduled position. We seek a replacement in the list in Step 18. If such an item exists, the size of list remains constant, and otherwise, it shrinks by 1.

Each value p_i and f_i can be computed in $O(\log n)$ steps by binary search, for a total of $O(n \log n)$ operations. The sorting operation for computing F takes $O(n \log n)$ time as well. The operations in the first while-loop are $O(n)$ in complexity. In the second while-loop, we must potentially delete an item from list (which can be done in constant time), find item k in Step 18 (which requires $O(\log n)$ steps), and perhaps insert an element back into list (which can be done in constant time). However, if a position index is kept explicitly at each iteration, the algorithm requires $O(n)$ operations at each update. Instead, in Step 15, we find the position of j in list via binary search. (Note that items appear in list in the same order that they appeared in F; furthermore, the tie-breaking criteria present in our sorting of F ensures that the sorting operation yields a unique sequence F. These facts permit us to execute binary search on list.) Step 15 therefore also takes $O(\log n)$ steps, and so the second while-loop requires $O(n \log n)$ steps. Finally, recovering the sequence at the end of the algorithm is an $O(n)$ operation, and so the overall complexity of this algorithm is $O(n \log n)$.

To illustrate this process, consider the $n = 10$ example whose v-, t-, p-, and f-values are depicted in Table 1. Initially, we have

$$F = \{8, 6, 9, 7, 5, 10, 3, 2, 4\} \text{ and}$$
$$\text{list} = \{9, 7, 10, 3, 2, 4\},$$

where items 8 and 6 do not belong to list because $f_8 = f_6 = 0$, and item 5 does not belong to list because it would be the third item, and $f_5 = 2 < 3$. Following the creation of list, $F = \{8, 6, 5\}$. We try to find a sequence such that item 1 can be sold, but this is impossible because $1 + |\text{list}| = 7$ and $p_1 = 10$. That is, even if every item in list is ordered before 1, the customer would purchase any item (other than 1) in the seventh position.

Table 1. Seller's problem example.

	1	2	3	4	5	6	7	8	9	10
v_i	30	68	78	60	83	85	83	92	84	81
t_i	86.1	85.0	83.6	82.0	80.0	77.5	74.2	69.5	62.5	50.0
pi	10	8	5	9	3	1	3	1	2	4
f_i	9	7	4	8	2	0	2	0	1	3

Having ruled out selling item 1, the seller attempts to sell item 2. First, item 2 is removed from list, and is not replaced: item 2 sits in the fifth position of list, and no item in F has an f-value of at least 5. Hence, item 2 could only be sold if $p_2 \leq 2 + |\text{list}| = 7$ (representing the 5-item sequence of list being scheduled, followed by item 1, followed by item 2), but $p_2 = 8$ and the item cannot be sold.

Turning our attention to selling item 3, we remove item 3 from list (again without replacement), and test $p_3 \leq 3 + |\text{list}| = 7$. This time, the relationship holds, and item 3 can indeed be sold to the customer. According to Proposition 1 and Lemma 1, one optimal sequence begins with

sequence: 9 7 2 1 3,

with the remaining five items being scheduled arbitrarily.

3. Max-Min Problem

The stopping problem we discuss in this paper essentially encompasses three different parameter sets: profits, thresholds, and customer values. A vital assumption that we made in Section 2 is that all parameters are known with certainty, and that the customer will employ an optimal strategy to select an item. However, in a more realistic setting, there may be some degree of uncertainty about the parameters and (especially in the case of a human decision maker) about the strategy that the customer uses. The seller should therefore incorporate knowledge of this uncertainty into the sequence. We consider in this section a conservative seller, who seeks a sequence that maximizes profit in a worst-case scenario. (The case of a seller who wishes to maximize expected profit instead given a particular data distribution is explored in Section 4.)

Note that it is not generally possible to specify a single "worst-case scenario" (i.e., an outcome of random data that is most damaging) for the seller. However, given an established sequence of items, it is possible to determine a worst-case set of data that results in the minimum profit for the seller. Hence, in this modeling strategy, we define an uncertainty set U of all possible combinations of threshold and customer values (with profit values being deterministic). In general, the only assumptions that we make regarding the structure of U is that it is nonempty, and the threshold values $\bar{t}$ that belong to U must satisfy $\bar{t}_1 \geq \cdots \geq \bar{t}_n > t_{n+1} = -\infty$. Note that from a game-theoretic perspective, one can view U as the set that defines boundedly rational behavior from the customer.

Denote $\ell(x, U)$ as the minimum profit possible given a sequence of items x over all possible data outcomes in U. The problem considered in this section seeks to maximize $\ell(x, \mathrm{U})$ over all permutations of items x, which is why we refer to this problem as a max-min optimization problem. This modeling philosophy is exactly that embodied by the *robust optimization* community; we refer the reader to Ben-tal, El Ghaoui, and Nemirovski (2009) for a thorough mathematical programming discussion of robust optimization.

Because identifying the worst-case data scenario corresponding to any item sequence x is an optimization problem, it is convenient to envision a third-party *adversary* who seeks the worst possible data outcome in U, given the seller's sequence of items. Hence, we now examine this problem as a Stackelberg game in which the seller arranges the items to be sold in some sequence, the adversary manipulates data (within the allowable uncertainty set U) , and the customer follows the previously stated optimal stopping strategy for selecting an item, but based on the parameters manipulated by the adversary.

A simple generalization of Algorithm 1 is not sufficient to solve the max-min problem described above. (Indeed, Corollary 1 in Section 4 will demonstrate that this problem is NP-hard in general.) However, we illustrate in this section one strategy for solving a max-min problem given a specific class of U-sets. Consider any set $\overline{U}$ in which the threshold values are fixed at their optimal t-values as computed by equations (1a) and (1b), $\forall i = 1, \ldots, n$, and where the item values v are restricted as follows for some nonnegative integer K:

$$v_i^{\prime} - \triangle y_i \leq v_i \leq v_i^{\prime} + \triangle y_i \quad \forall i = 1, \ldots, n \tag{5a}$$

$$\sum_{i=1}^{n} y_i \leq K \tag{5b}$$

$$y_i \in \{0,1\} \quad \forall i = 1, \ldots, n \tag{5c}$$

where $\triangle \geq 0$. Here, $v_i^{\prime}$ acts as a nominal value for each $i = 1, \ldots, n$. Note that constraints (5a) state that the true value for item is somewhere in the interval $[v_i^{\prime} - \triangle, v_i^{\prime} + \triangle]$. Constraint (5b) states that only some K parameters v_i may deviate from their nominal values, and (5c) states logical restrictions on the y-variables that control which parameters deviate from their nominal values.

Our approach to solve this problem follows the general strategy employed in Algorithm 1: the seller searches for a sequence that results in some item $i = 1, \ldots, n$ being chosen by the customer, stopping when the highest-profit item can be sold. However, we must now take the adversary's role into account, noting that the customer values may be modified from their nominal values by the adversary to prevent an item from being sold, or induce a (low-profit) item to be sold.

For convenience, we define:

$$J_i^1 = \{1, \ldots, i-1\} \quad \forall i = 1, \ldots, n, \tag{6a}$$

$$J_i^2 = \{i+1, \ldots, n\} \quad \forall i = 1, \ldots, n. \tag{6b}$$

Also, given a sequence of items, define pos_i as the position of item i in the sequence.

We next provide the following proposition, which establishes the form of optimal sequences given an uncertainty set of the form $\overline{U}$.

Proposition 2. Let i be the (smallest) index of the item sold to the customer in an optimal sequence given an uncertainty set of the form $\overline{U}$. An optimal sequence consists of items in sets $A_1, \ldots, A_5$ (some of which may be empty) in the order

$$A_1 - A_2 - A_3 - A_4 - \text{item } i - A_5 \tag{7}$$

such that:

- A_1 consists of items $j \in J_i^2$ such that $v_j' + \triangle < t_{pos_j+1}$,
- A_2 consists of items $j \in J_i^1$, K of which satisfy $v_j' > t_{pos_j+1}$, including the last elementof A_2,
- A_3 consists of items $j \in J_i^2$ such that $v_j' < t_{pos_j+1}$,
- A_4 consists of items $j \in J_i^1$, and
- A_5 consists of items $j \in J_i^2$,

where sets $A_1, \ldots, A_5$ and $\{i\}$ form a partition of the items $\{1, \ldots, n\}$.

Proof. To begin, it is useful to interpret the sets A_i, for $i = 1, \ldots, 5$.

- A_1 consists of items $j \in J_i^2$, which can safely be scheduled so that even if the adversary modifies v_j to take on its largest possible value, the customer will not purchase it.
- A_2 consists of items $j \in J_i^1$ that (by assumption) cannot be sold. Indeed, the adversary has been forced to use all K of its allotted value deviations (i.e., setting $y_i = 1$ for items $i \in A_2$) to prevent the customer from buying these items. In particular, the adversary must have modified the last item's value in A_2 to prevent it from being sold.
- A_3 is similar to A_1, with the exception that these items will not be sold if the adversary cannot modify their values (instead of the stronger condition stated for items in A_1). The placement of these items before item i in the sequence is valid because the adversary used all K modifications to prevent items in A_2 from being sold. Hence, if A_2 is empty, then so is A_3.
- A_4 is similar to A_2, without the caveat that the adversary must take action to prevent their sale.
- A_5 consists of all items in J_i^2 not contained in A_1 or A_3.

Consider any original sequence in which item i can be sold, which does not satisfy (7). We show that there also exists a modified sequence satisfying (7) such that item i is still sold.

First, consider the case in which the adversary is forced to use K modifications to prevent an item in J_i^1 from being sold in the original sequence, before item i is eventually sold. (For simplicity, we say that the adversary has "attacked" an item in J_i^1 if the adversary sets $v_i = v_i' - \triangle$ to prevent it from being sold.) In the original sequence, let p' be the position of the earliest scheduled item that belongs to J_i^1, and let p'' be the position of the Kth item that is attacked in J_i^1, noting that $p'' < pos_i$. Suppose that an item $j \in J_i^2$ is positioned so that $p' < pos_j < p''$. Note that j satisfies the criterion $v_j^{\prime} + \triangle < t_{pos_j+1}$, because otherwise, the adversary (not having used all K attacks before position pos_j) could induce the customer to purchase item j. We could therefore swap the position of item j with the item in J_i^1 in slot p', with item i still being sold. Note that j must still satisfy $v_j^{\prime} + \triangle < t_{p'+1}$ because $t_{p'+1} > t_{pos_j+1}$; the latter inequality also indicates that if the adversary had to attack the item in slot p' in the original sequence, it must still do so when this item is in the later slot pos_j in the modified sequence. After repeating this procedure, all items in A_1 will precede those in A_2.

Furthermore, suppose that the last item in A_2 is not attacked by the adversary. By swapping the order of any item j in A_2 that is attacked with the last item in A_2, item j is still attacked in its later slot. Hence, a sequence exists in which the last item in A_2 is attacked.

Now, say that the last item in A_2 ends at slot q' in the original sequence. If no items in J_i^2 are scheduled after A_2 and before i, then A_3 is empty. Else, suppose that the last item in J_i^2 scheduled before item i is in position q''. If there is no item in J_i^1 positioned in some slot q such that $q' < q < q''$, then the schedule follows (7) as desired. Else, consider such an item j in position q. Suppose that we modify the sequence by swapping the position of items in positions q and q''. The item formerly in position q'' belongs to J_i^2 and cannot be sold at the earlier slot, q. Item j belongs to J_i^1 and cannot be sold by assumption (and is not attacked in the original or modified sequence because it follows the set of items A_2). Hence, item i will still be sold in the modified sequence. Performing all such swaps yields a sequence that satisfies the conditions stated in (7) pertaining to items in slots $1, \dots, pos_i$.

To verify that $A_5 \subset J_i^2$, suppose that an item $j \in J_i^1$ is scheduled after i in the original sequence. A modified sequence would place j in the position of item i, and shift all items placed in slots $pos_i, \dots, pos_j - 1$ back one spot in the modified sequence. Because item j cannot be selected by the customer, and because item i is appearing later in the modified sequence than in the original sequence (where it was selected), item i will still be sold in the modified sequence. Hence, a sequence exists in which A_5 consists only of items in J_i^2.

Second, we consider the case in which the adversary does not need to use all K modifications in the original sequence to prevent items in J_i^1 from being sold. If the original sequence does not satisfy (7), then there exists an item $j \in J_i^1$ scheduled before an item $k \in J_i^2$, which is scheduled before i. By the same argument as above, items j and k can be swapped in a modified sequence, and item i would still be sold in the modified sequence. Also, the same argument showing that A_5 consists only of items belonging to J_i^2 holds as above. Repeating these swaps leads to a sequence that satisfies (7), with A_2 and A_3 being empty. □

The following lemmas allow us to further restrict our attention to special sequences that obey (7).

Lemma 2. Consider an optimal sequence of the form (7) in which $A_2 \neq \emptyset$, $|A_1| = p$ for some non-negative integer p, and item i is sold to the customer. Let the items in J_i^1 be sorted in non-increasing order of their nominal values, yielding the sequence $\alpha(1), \ldots, \alpha(|J_i^1|)$. Also, let J_i^2 be sorted in non-increasing order of their nominal values, yielding the sequence $\beta(1), \ldots, \beta(|J_i^2|)$. Then:

1. Items in A_1 are ordered in the following greedy fashion: Set $j = 1$ and $q = 1$, and determine if $v'_{\beta(j)} + \triangle < t_{q+1}$. If so, then place $\beta(j)$ in position q and increment q by 1. In either case, increment j by 1. Stop if $q = p + 1$; else, repeat.
2. Items $\alpha(1), \ldots, \alpha(K)$ are scheduled so that the adversary attacks each item to prevent them from being sold. Item $\alpha(1)$ is placed in the earliest position $q >$ p such that $v'_{\alpha(1)} > t_{q+1}$. Then, item $\alpha(j)$ is placed in the earliest position $q > pos_{\alpha(j-1)}$ such that $v'_{\alpha(j)} > t_{q+1}$, for each $j = 2, \ldots, K$. All other positions $p + 1, \ldots, pos_{\alpha(K)}$ that have not been assigned an item (if any) are assigned unscheduled items from J_i^1 in any arbitrary order.
3. Items in A_3 are ordered according to the same greedy algorithm as for A_1, except we start at position $q = |A_1| + |A_2| + 1$, ignore those items that have been scheduled in A_1, and test whether $v'_{\beta(j)} < t_{q+1}$, noting that the adversary cannot adjust the values of items in A_3.

Proof. Consider any original sequence of the form (7) in which $A_2 \neq \emptyset$, $|A_1| = p$, and item i is sold to the customer, but was not generated according to Claims 1-3 in the lemma. We again show that a modified sequence exists in which item i is sold that does conform to these three claims.

First, suppose that Claim 1 does not hold in the original sequence. Then there exists a minimum index p', with item $j \in J_i^2$ positioned in p', such that a higher-valued item $k \in J_i^2$, $v'_k + \triangle < t_{p'+1}$, is positioned in slot $p'' > p'$. We could generate a modified sequence by swapping the position of items j and k. By assumption, item k would not be purchased by the customer (even if its value were modified by the adversary). If $p'' < pos_i$, then k could not have been purchased by the customer in position p'', and thus j also could not be purchased in that position because $v'_j < v'_k$. If $p'' > pos_i$, then item i is purchased before item j would be encountered by the customer in the modified sequence. By performing all such swaps, we recover a modified sequence in which Claim 1 holds true. Note that Claim 3 also holds true by the same argument.

To show that Claim 2 holds true, we use similar mechanics as in the proof that Claims 1 and 3 hold true. Let p' be the minimum index in A_2, with item $j \in J_i^1$ again being positioned in slot p', such that item j is attacked by the adversary in the current sequence (e.g., $v'_j > t_{p'+1}$) , and where a higher-valued item $k \in J_i^1$ is positioned in slot $p'' > p'$. Consider a modified sequence obtained by swapping the position of items j and k. Item k must be attacked by the adversary in the modified sequence because $v_k > v_j$. Note that item j must still be attacked in the modified sequence as well, because (a) it was attacked in the original sequence in position p', (b) item j is moved to a later slot p'', and (c) by the sequence rule (7), we have that $p'' < pos_i$, and the adversary must attack item j to prevent it from being sold before i. Performing all such swaps gives us a sequence that satisfies Claim 2. □

Lemma 3. Consider an optimal sequence of the form (7) in which $A_2 = A_3 = \emptyset$. Then there exists an optimal sequence that maximizes $|A_1|$, with items in A_4 being arbitrarily ordered. Moreover, items in A_1 can be ordered by the same greedy approach given in the first claim of Lemma 2.

Proof. Suppose that item i is a best-profit item that can be sold in any optimal solution, and consider any sequence that results in item i being sold, does not force the adversary to make K attacks on J_i^1 items, and does not maximize the cardinality of A_1. By increasing the number of items in A_1, and retaining the same sequence of items in A_4 (which includes all items in J_i^1), we have that item i is pushed back to a later position in the sequence. Note that no items in A_1 can be purchased by the customer (by construction), and that it is impossible to sell items in A_4 (by assumption). Item i must therefore be sold in this later position. The fact that items in A_4 can be arbitrarily ordered is due to the fact that A_2 is empty, which implies that the adversary has made fewer than K attacks. Thus, there is no implicit requirement to order items in A_4 that forces the adversary to attack. The justification for the greedy algorithm used to arrange items in A_1 is the same as given in Lemma 2. □

Observe that it is difficult to predict before solving a problem whether an optimal sequence will be generated according to Lemma 2 or 3, and if the former, which value of p $(= |A_1|)$ should be used. Therefore, an algorithm used to solve the case in which our uncertainty set is of the form $\overline{U}$ will consider each possibility allowed by Lemmas 2 and 3.

To state this algorithm, when trying to sell item $i = 1, \ldots, n$ after establishing that items $1, \ldots, i-1$ cannot be sold, we sort items in J_i^1 and J_i^2 in non-increasing order of their nominal values, to yield the sequences $\alpha(1), \ldots, \alpha(|J_i^1|)$ and $\beta(1), \ldots, \beta(|J_i^2|)$, respectively. We first attempt to order the items according to Lemma 3. If item i cannot be sold by this sequence, we set an integer parameter $P = |A_1|$ in the sequence produced by Lemma 3, and attempt to create a sequence generated according to Lemma 2 for each $p = 0, \dots, P$. (Note that there may not be a sequence possible for some values of p: if p is too small, then there may not be enough items in J_i^1 to fill all slots between slot $p + 1$ and $pos_{\alpha(K)}$, where the last item in A_2 must be scheduled.) If no sequence can be found that sells item i, then we set $i = i + 1$ and restart the process. The algorithm ends as soon as a sequence is found that sells item i.

Note that this algorithm can be performed in $O(n^3)$ steps, given by the $O(n^2)$ complexity of searching through sequences produced by Lemmas 2 and 3 in trying to sell item i, and the $O(n)$ number of items i that must be explored by the algorithm. We leave for future research the exploration of a more sophisticated algorithm that would attempt to reduce the effort required to search the sequences given by Lemmas 2 and 3.

4. Maximization of Expected Profit

In this section, we examine an alternative characterization of uncertainty that arises in the seller's problem. Here, the profit, value, and customer threshold data are all potentially uncertain. We model this uncertainty by considering a set Q of scenarios, where scenario $q \in Q$ occurs with probability π^q, and in which all profit, value, and threshold data associated with scenario q are superscripted by q (e.g., v_i^q reflects the value of item i in scenario q, and so on). Denote $\pi^q \geq 0$ as the probability of realizing scenario $q \in Q$, where $\sum_{q \in Q} \pi^q = 1$.

Unlike in Section 3, we instead examine the problem of maximizing expected profit (which we call problem **EXP**), rather than maximizing the minimum profit that could be obtained from a sequence. More formally, problem EXP seeks a sequence of all items, such that the expected profit (given by the summation over all $q \in Q$ of the item's profit that would be sold in scenario q multiplied by π^q) is maximized. The following theorem and its corollary state that once some degree of uncertainty is incorporated into the problem studied in Section 2 (whether for the max-min or max-expectation case), the problem can become substantially difficult in general.

Theorem 1. Problem EXP is NP-hard, even when all profit values are deterministic and the customer uses optimal threshold values.

Proof. The corresponding decision problem, EXPD, is stated as follows: for a given parameter G, does there exist a sequence whose expected profit is at least G? We will show that EXPD is NP-complete, which implies that EXP is NP-hard. Assuming that the customer solves the stopping problem in polynomial time, EXPD clearly belongs to NP: for any given sequence we can easily determine the item that is chosen by the customer in each scenario, compute the expected profit, and check to see if the expected profit is at least G.

Next we show that Expd is NP-complete by transforming the classical 3SAT problem to an equivalent instance of Expd. First, we define 3SAT as follows (Garey & Johnson, 1979).

> Consider a set of clauses C on a set $U = \{u_1, \ldots, u_n\}$ of n binary variables (which can take values of either true or false). Each clause contains three "literal" values, each of which is a true or false value for one particular variable. Does there exist a "truth assignment" of binary values to the variables in U, that is, an assignment of true or false values for every variable in U such that every clause has a literal that matches some value in the assignment?

We denote u_i^T (u_i^F) as a literal having a true (false) value for variable i. For instance, if a clause consists of literals $\{u_1^T, u_3^F, u_6^T\}$, then any truth assignment must satisfy the condition that either u_1 = true, or u_3 = false, or u_6 = true.

Consider any 3SAT instance with n variables and m clauses, denoted by $C_j, \forall j = \{1, \ldots, m\}$. We define items $T_i, \forall i = \{1, \ldots, n\}$, corresponding to u_i^T literals and $F_i, \forall i = \{1, \ldots, n\}$, corresponding to u_i^F literals in our Expd instance. Each of these $2n$ items has a profit of 1. We define one further item, denoted by Z, which has a profit of 0. Let the customer's threshold values be optimal for her, as computed in (1a) – (1b). Also, we define parameter t^* to be a value satisfying the relationship $t_{n+1} \le t^* < t_{n+2}$.

Now let scenario set $Q = \{1, \ldots, n + m\}$, and define the customer values for each scenario as follows:

$$v_Z^q = 100 \quad \forall q = 1, \ldots, n+m, \tag{8a}$$

$$v_{T_q}^q = t^* \quad \forall q = 1, \ldots, n, \tag{8b}$$

$$v_{F_q}^q = t^* \quad \forall q = 1, \ldots, n, \tag{8c}$$

$$v_w^q = t^* \quad \forall q = n+1, \ldots, n+m, \text{ item } w \text{ corresponds to a literal in } C_{q-n} \tag{8d}$$

$$v_i^q = 0 \quad \text{otherwise.} \tag{8e}$$

Note that in (8d), the index w refers to the item T_i (F_i), if the literal u_i^T (u_i^F) is a member of the clause C_{q-n}. Finally, $\pi^q = 1/(n + m), \forall q \in Q$, and $G = 1$.

To prove that the Expd instance is equivalent to the 3SAT instance, we show there exists a 3SAT solution if and only if there also exists a solution to Expd. As a preliminary note, for any sequence we define *early items* as the first n items in the sequence and *late items* as the next n items (but not the last item in slot $2n + 1$).

First, suppose that there exists a solution to the 3SAT instance. To construct a sequence, we place item F_i in slot i and item T_i in slot $n + i$ if the 3SAT variable u_i is true, $\forall i = 1, \ldots, n$. Otherwise, if u_i is false, we place item T_i in slot i and item F_i in slot $n + i$, $\forall i = 1, \ldots, n$. Item Z is positioned in slot $2n + 1$. Note that in any scenario, an early item will not be chosen, because the early item values equal either t^* or 0, which are less than t_{n+2} and will not be chosen by the customer when they belong to the first n positions of the sequence. If a late item exists having a value of t^* in scenario q, the customer will buy one such item at a profit of 1 to the seller in scenario q. For scenario $q = 1, \ldots, n$, note that either T_q or F_q is a late item, and has value t^* in scenario q. For scenario $q = n + 1, \ldots, n + m$, one of the three items in clause $q - n$ corresponds to a late item having value t^*, due to the assumption that the late items correspond to a 3SAT truth assignment. In every scenario, a profit of 1 is obtained, and so the expected profit is $G = 1$. Hence, the Expd instance has a solution.

Next, suppose that there exists a solution to the transformed Expd instance. Note that this is equivalent to enforcing the condition that item Z is *not* chosen by the customer in any scenario. We will show that the late items in such a sequence correspond to a solution to the 3SAT instance.

Observe that item Z must be placed in slot $2n + 1$. If not, some item T_i (or F_i) must be the last item in the sequence. If its complementary item F_i (T_i) is an early item, the customer skips the early item in scenario i and chooses item Z. Otherwise, item F_i (T_i) is a late item, which implies there exists a pair of items T_k and F_k for some k that are both scheduled as early items. This results in item Z being chosen in scenario k. Hence, item Z must be positioned in slot $2n + 1$ in our Expd solution in order to avoid selling item Z in the first n scenarios. Moreover, by the same logic, exactly one of the items T_i or F_i must be a late item, for $i = 1, \ldots, n$.

Using the above observations, we set the variable u_i to true (false), if item T_i (F_i) is a late item. Because Z is not chosen in any scenario $q = n + 1, \ldots, n + m$, a late item corresponds to a literal in clause C_j for each $j = 1, \ldots, m$. Therefore, the proposed 3SAT solution is feasible to the 3SAT instance.

Finally, note that the transformation creates a polynomial number of items and scenarios. It is hence a polynomial transformation if t^* is polynomially representable (i.e., if we require that the number of bits required to represent t^* is polynomially bounded by n and m). For expediency, we could assume that the customer uses thresholds with precision bounded by a polynomial function of n. Taking $t^* = (t_{n+1} + t_{n+2})/2$, an encoding of t^* can also performed using a polynomial number of bits. More generally, even if the customer uses exact thresholds with unlimited precision, we demonstrate in the Appendix that a value for t^*, encoded using a polynomial number of bits, can be computed. This completes the proof. □

Indeed, a consequence of this theorem is that the general max-min problem (for any arbitrary $U \neq \emptyset$ with monotone threshold values) must also be NP-hard.

Corollary 1. The max-min problem is NP-hard for general uncertainty sets U, even when all profit values are deterministic and the customer uses optimal threshold values.

Proof. We can use the same transformation given for Theorem 1, where U consists of the discrete value sets given above with the threshold values determined by (1a) and (1b). We would then insist on a worst-case profit of 1, which turns out to be identical to requiring that the expected profit equals 1. Thus, the max-min problem with general uncertainty sets is also NP-hard. Observe that we do not make any claims regarding the inclusion of a decision variant of the max-min problem in the class NP, because it is not clear that solving the adversary's problem is generally achievable in polynomial time for general U. □

The implication of Theorem 1 is that no polynomial-time solution exists for problem EXP (unless P = NP). We thus provide a mixed-integer programming (MIP) model for this problem, which can be solved by standard MIP techniques (Nemhauser & Wolsey, 1999). (Indeed, stochastic programming models like the one we face in this section are often solved by decomposition techniques [Birge & Louveaux, 1997], but the development of these algorithms is beyond the scope of our paper.)

To formulate this MIP, we let $N = \{1, \ldots, n\}$ for convenience, and define the following set of decision variables. Let x_{ij}, $\forall i \in N, j \in N$, be a binary decision variable that equals 1 if item i is in slot j of the sequence and 0 otherwise. Also, let z_j^q, $\forall j \in N$, $q \in Q$, be a binary decision variable that equals 1 if the item in slot j is chosen by the customer in scenario q, and 0 otherwise. We define a new parameter a_{ij}^q, $\forall i \in N, j \in N, q \in Q$, which equals 1 if item i could possibly be chosen by the customer in scenario q if placed in slot j, that is:

$$a_{ij}^q = \begin{cases} 1 & \text{if } i \in N,\ j \in \{p_i, \ldots, n\},\ q \in Q, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

We begin by stating a nonlinear MIP that models problem EXP:

$$\max \sum_{q \in Q} \neq^q \sum_{i \in N} b_i \sum_{j \in N} x_{ij} z_j^q \tag{10a}$$

$$\text{s.t.} \sum_{j \in N} x_{ij} = 1 \quad \forall i \in N, \tag{10b}$$

$$\sum_{i \in N} x_{ij} = 1 \quad \forall j \in N, \tag{10c}$$

$$z_j^q \geq a_{ij}^q x_{ij} - \sum_{k=1}^{j-1} z_k^q \quad \forall i \in N,\ j \in N,\ q \in Q, \tag{10d}$$

$$z_j^q \leq \sum_{i \in N} a_{ij}^q x_{ij} \quad \forall j \in N,\ q \in Q, \tag{10e}$$

$$z_j^q \leq 1 - \sum_{k=1}^{j-1} z_k^q \quad \forall j \in N,\ q \in Q, \tag{10f}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in N,\ j \in N, \tag{10g}$$

$$z_j^q \in \{0,1\} \quad \forall j \in N,\ q \in Q. \tag{10h}$$

The objective function (10a) calculates the expected profit: for each scenario $q \in Q$, if $z_j^q = 1$, then the seller receives a profit of b_i, weighted by probability π^q, if item $i \in N$ is placed in slot $j \in N$ (i.e., if $x_{ij} = 1$). Constraints (10b) and (10c) guarantee that each item is assigned to exactly one slot, and vice versa. Constraints (10d) to (10f) enforce the condition that z_j^q equals 1 if and only if j is the first slot for which the assigned item i satisfies the condition $v_i^q > t_{j+1}$ in scenario q. To see this, consider an item i positioned in slot j ($x_{ij} = 1$), and observe that if $v_i^q \leq t_{j+1}$, then $a_{ij}^q = 0$. Thus, (10e) implies that $z_j^q = 0$ in this case. If, however, $v_i^q > t_{j+1}$, and thus $a_{ij}^q = 1$, then there are two cases to consider. If there does not exist a slot before j whose corresponding item has been chosen (e.g., $\sum_{k=1}^{j-1} z_k^q = 0$), then (10d) forces $z_j^q = 1$. If there does exist a $k \in \{1, \ldots, j-1\}$ such that $z_k^q = 1$, then (10f) forces $z_j^q = 0$ as desired. Observe therefore that constraints (10h) can be replaced simply with $z_j^q \geq 0$, $\forall j \in N, q \in Q$, noting that z-variables must be binary-valued given binary x-values.

Although the objective function is nonlinear, it can be easily converted to a linear function due to the facts that (a) nonlinearity only arises due to the nonlinear terms $x_{ij} z_j^q$, and (b) the x-variables are restricted to be binary-valued. By introducing a new set of variables ψ_{ij}^q, $\forall i \in N, j \in N, q \in Q$, which are designed to take on the value of $x_{ij} z_j^q$, we obtain the following linear MIP:

$$\max \sum_{q \in Q} \pi^q \sum_{i \in N} b_i \sum_{j \in N} \psi_{ij}^q \tag{11a}$$

s.t. constraints (10b) to (10h), (11b)

$$\psi_{ij}^q \leq x_{ij} \quad \forall i \in N, j \in N, q \in Q, \tag{11c}$$

$$\psi_{ij}^q \leq z_j^q \quad \forall i \in N, j \in N, q \in Q, \tag{11d}$$

Observe that (11c) and (11d) force $\psi_{ij}^{q} \le x_{ij} z_{j}^{q}$; equality comes from the fact that optimization will force the ψ-variables to take on their largest permissible values.

5. Conclusions and Future Research

In this paper, we addressed a finite-horizon optimal stopping problem from the seller's perspective. We began by demonstrating that when the customer is optimal, the seller can optimize profit from selling items in $O(n \log n)$ time, where n is the number of items for sale. The vast literature in experimental research on stopping problems has shown that human decision makers, acting as the customers, tend to stop searching too soon, and in any case cannot be assumed to be optimal decision makers. We model the unpredictability of human decision-making behavior by analyzing situations in which the customer's values and/or her or his stopping thresholds are uncertain. We first examined a max-min case in which the seller wishes to maximize the minimum profit that can be made given some uncertainty set in which the data values must reside. A special case of this max-min problem that we studied here remains polynomially solvable. Next, we examined the case in which the seller wishes to maximize expected profit. This problem turns out to be NP-hard, even when uncertainty is confined to the customer values.

We provided a formulation for solving the problem of maximizing expected profit (in which uncertainty can be applied to any part of the data except for n). However, we did not explore solution techniques tailored for this problem, beyond the use of standard mixed-integer programming solvers. When n or $|Q|$ is large, it is not likely that formulation (11) will be tractable. One area of future research may instead focus on custom solution techniques for solving (11) within reasonable computational limits. Another area of interest is certainly in laboratory testing of these models. Conservative models (such as those presented in Section 3) tend to sacrifice potential profit in favor of guaranteeing minimum profits. It would be of interest to demonstrate how conservative the seller should be in practice given a human decision maker. Furthermore, we have assumed that the items' values and profits are independent in general. As an extension to our work, the scenarios in which there exist a degree of correlation between values and profits can be also be considered for future studies. Finally, an expanded version of this problem may attempt to observe this game in a repeated setting, in which the customer adapts the purchasing strategy based on the tendencies of a profit-motivated seller.

References

Bartoszynski, R., & Govindarajulu, Z. (1978). The secretary problem with interview cost. *Sankhya: The Indian Journal of Statistics, 40,* 11–28.

Bearden, J. N., & Rapoport, A. (2005). Operations research in experimental psychology. In J. C. Smith (Ed.), *Tutorials in Operations Research: Emerging Theory, Methods, and Applications* (Vol. 1, pp. 213–236). INFORMS.

Bearden, J. N., Rapoport, A., & Murphy, R. O. (2006). Sequential observation and selection with rank-dependent payoffs: An experimental test. *Management Science, 52,* 1437–1449.

Bellman, R. (1957). *Dynamic Programming.* Princeton University Press.

Ben-Tal, A., El Ghaoui, L., & Nemirovski, A. (2009). *Robust Optimization.* Princeton Series in Applied Mathematics. Princeton University Press.

Birge, J. R., & Louveaux, F. (1997). *Introduction to Stochastic Programming*. Springer-Verlag.

Chow, Y. S., Robbins, H., & Siegmund, D. (1971). *Great Expectations: The Theory of Optimal Stopping.* Houghton Mifflin.

Ferguson, T. S. (1989). Who solved the secretary problem? *Statistical Science, 4,* 282–296.

Ferguson, T. S. (2010). Optimal stopping and applications. http://www.math.ucla.edu/~tom/Stopping/Contents.html.

Freeman, P. R. (1983). The secretary problem and its extensions: A review. *International Statistical Review, 51,* 189–206.

Gardner, M. (1960). Mathematical games. *Scientific American, 202,* 150–153.

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman.

Gilbert, J., & Mosteller, F. (1966). Recognizing the maximum of a sequence. *Journal of the American Statistical Association, 61,* 35–73.

Lee, M. D. (2006). A hierarchical Bayesian model of human decision-making on an optimal stopping problem. *Cognitive Science, 30*(3), 555–580.

Nemhauser, G. L., & Wolsey, L. A. (1999). *Integer and Combinatorial Optimization*. John Wiley and Sons.

Pressman, E. L., & Sonin, I. M. (1972). The best choice problem for a random number of objects. *Theory of Probability and Its Applications, 17,* 657–668.

Samuels, S. M. (1991). Secretary problems. In B. K. Gosh & P. K. Sen (Eds.), *Handbook of Sequential Analysis* (pp. 381–405). Marcel Dekker.

Seale, D. A., & Rapoport, A. (1997). Sequential decision making with relative ranks: An experimental investigation of the 'secretary problem. *Organizational Behavior and Human Decision Processes, 69,* 221–236.

Smith. A. (1975). Secretary problem with uncertain employment. *Journal of Applied Probability, 12,* 620–624.

Appendix: Representation of Threshold Values

In order to precisely discuss the complexity of the problems under investigation here, we must address the size of the data used in our computations. Even after making the simplifying assumption that the customer's values are uniformly distributed on the interval [0,100], it is not clear that the customer can truly solve the optimal purchasing (stopping) problem in polynomial time. The recursions in (1a) and (1b) allow the generation of threshold data in $O(n)$ time provided that computations are performed in constant time. However, note that (after dividing the maximum customer values by 100) $t_n = 1/2$, and that $t_i = (t_{i+1}^2 + 1)/2$ for each $i = 1, \ldots, n-1$. This means that $t_{n-1} = 5/8$, $t_{n-2} = 89/128$, and so on. The implication is that $t_{n-j+1} = \alpha_j/(2^{2^j-1})$ for some integer numerator α_j, $\forall j = 1, \ldots, n$. Unfortunately, this implies that the number of bits required to store α-values is $O(2^n)$. Therefore, it is not technically permissible to let $t^* = (t_{n+1} + t_{n+2})/2$ in the proof of Theorem 1, because storing this value evidently requires an exponential number of bits. (In fact, it is more accurate to say that we do not know how to store this number using a polynomial number of bits.)

A simplifying assumption would state that the customer makes all computations with finite precision, and that this level of precision is treated as a constant value in our computational analysis. But interestingly, for the case in which the customer perceives a uniform distribution of probability data, Theorem 1 holds true even when no assumption is made that restricts the precision of the customer's computations. We discuss the details of this argument below.

Consider the customer's optimal stopping problem with n total items. We seek a sequence of values $s_1, \ldots, s_n$ such that $t_n = 0.5 \le s_n < t_{n-1} \le s_{n-1} < \cdots < t_1 \le s_1$, where $s_{n+1-j} = \beta_j/2^{j+2}$ for $j = 1, \ldots, n$, such that each β_j is a positive integer and can thus be represented using no more than $n + 2$ bits. It is easy to show that $\beta_1 = 4$, $\beta_2 = 10$, $\beta_3 = 23$, and $\beta_4 = 48$ are valid in the sense that they generate s-values that satisfy the inequality chain above, after dividing by 8, 16, 32, and 64, respectively. For $j = 5$, we can select β_5 to be any value in {100, 101, 102}. Using $j = 5$ as a base case, we will prove by induction that for any $j \ge 5$, there exist at least three values of β_j such that $s_{n+1-j} = \beta_j/2^{j+2}$ is valid.

Suppose that this property holds for a given $j \ge 5$. We thus have:

$$t_{n+1-j} \le \beta_j/2^{j+2} < (\beta_j + 2)/2^{j+2} < t_{n-j},$$

for some β_j. Consider now the computation of a threshold value β_{j+1} via the recursion (1b). Note that a threshold τ'_{j+1} based on the value $\beta_j/2^{j+2}$ would be computed as $(\beta_j^2 + 2^{2(j+2)})/2^{2(j+2)+1}$, and a threshold τ''_{j+1} based on the value $(\beta_j + 2)/2^{j+2}$ would be computed as $(\beta_j^2 + 4\beta_j + 4 + 2^{2(j+2)})/2^{2(j+2)+1}$. However, to use these thresholds for s-values, the denominator must be no more than 2^{j+3} rather than $2^{2(j+2)+1}$. Hence, consider the following threshold values, obtained after dividing the numerator and denominator by 2^{j+2} and rounding the numerator so that it is integer-valued:

$$\tau'_{j+1} = \frac{\beta'_{j+1}}{2^{j+3}} = \frac{\left\lceil \left(\beta_j^2 + 2^{2(j+2)}\right)/2^{j+2} \right\rceil}{2^{j+3}} \tag{12}$$

$$\tau''_{j+1} = \frac{\beta''_{j+1}}{2^{j+3}} = \frac{\left\lfloor \left(\beta_j^2 + 4\beta_j + 4 + 2^{2(j+2)}\right)/2^{j+2} \right\rfloor}{2^{j+3}}. \tag{13}$$

If $\beta'_{j+1} \le \beta''_{j+1}$, then β_{j+1} can validly take on any integer value in the interval $[\beta'_{j+1}, \beta''_{j+1}]$. We show here that $\beta'_{j+1} + 2 \le \beta''_{j+1}$. Note that when $j = 5$, we have that $t_{n-4} > 3/4$. Comparing the difference in the numerators in (13) and (12) before rounding, we have:

$$\frac{\left(\beta_j^2 + 4\beta_j + 4 + 2^{2(j+2)}\right) - \left(\beta_j^2 + 2^{2(j+2)}\right)}{2^{j+2}} = \frac{4\beta_j + 4}{2^{j+2}} > 3,$$

where the latter inequality is due to the fact that $\beta_j/2^{j+2} \ge t_{n-4} > 3/4$. Performing the ceiling and floor operations on the numerators of (12) and (13), respectively, narrows the gap between these values to at least 2, which verifies our claim. (Observe now that we have required $\beta'_j + 2 \le \beta''_j$ so that we are guaranteed to have a nonempty interval $[\beta'_{j+1}, \beta''_{j+1}]$ when using the above induction argument.)

Therefore, we can compute the β-values as given by the base cases above for $j = 1, \ldots, 5$, and then by recursion using (12) thereafter, using a polynomial number of bits. Hence, in the uniform distribution case, Theorem I is still valid even when the customer uses infinite precision, when we select $t^* = s_{n+1}$ in that transformation, assuming that $j \ge 5$ (with the case of $j \le 4$ being trivial). This guarantees that $t_{n+1} > t^* > t_{n+2}$, and that t^* is encodable using a polynomial number of bits.

Acknowledgments

The authors gratefully acknowledge the comments of Dr. Michael D. Lee, whose remarks led to an improved version of this paper. Dr. Smith gratefully acknowledges the support of the Air Force Office of Scientific Research under Grants FA9550-07-1-0404 and FA9550-08-1-0189, and of the Defense Threat Reduction Agency under Grant HDTRA1-10-1-0050.

Paper submitted: September 17, 2010
Paper accepted: December 12, 2010