ON-LINE DIFFERENCE MAXIMIZATION*

MING-YANG KAO† AND STEPHEN R. TATE‡

Abstract. In this paper we examine problems motivated by on-line financial problems and stochastic games. In particular, we consider a sequence of entirely arbitrary distinct values arriving in random order, and must devise strategies for selecting low values followed by high values in such a way as to maximize the expected gain in rank from low values to high values.

First, we consider a scenario in which only one low value and one high value may be selected. We give an optimal on-line algorithm for this scenario, and analyze it to show that, surprisingly, the expected gain is $n - O(1)$, and so differs from the best possible *off-line* gain by only a constant additive term (which is, in fact, fairly small — at most 15).

In a second scenario, we allow multiple nonoverlapping low/high selections, where the total gain for our algorithm is the sum of the individual pair gains. We also give an optimal on-line algorithm for this problem, where the expected gain is $n^2/8 - \Theta(n \log n)$. An analysis shows that the optimal expected off-line gain is $n^2/6 + \Theta(1)$, so the performance of our on-line algorithm is within a factor of 3/4 of the best off-line strategy.

1. Introduction. In this paper, we examine the problem of accepting values from an on-line source and selecting values in such a way as to maximize the difference in the ranks of the selected values. The input values can be arbitrary distinct real numbers, and thus we cannot determine with certainty the actual ranks of any input values until we see all of them. Since we only care about their ranks, an equivalent way of defining the input is as a sequence of n integers $x_1, x_2, \ldots, x_n$, where $1 \leq x_i \leq i$ for all $i \in \{1, \ldots, n\}$, and input x_i denotes the rank of the ith input item among the first i items. These ranks uniquely define an ordering of all n inputs, which can be specified with a sequence of ranks $r_1, r_2, \ldots, r_n$, where these ranks form a permutation of the set $\{1, 2, \ldots, n\}$. We refer to the r_i ranks as *final ranks*, since they represent the rank of each item among the final set of n inputs. We assume that the inputs come from a probabilistic source such that all permutations of n final ranks are equally likely.

The original motivation for this problem came from considering on-line financial problems [2, 4, 7, 8, 9], where maximizing the difference between selected items naturally corresponds to maximizing the difference between the buying and selling prices of an investment. While we use generic terminology in order to generalize the setting (for example, we make a "low selection" rather than pick a "buying price"), many of the problems examined in this paper are easily understood using notions from investing. This paper is a first step in applying on-line algorithmic techniques to realistic on-line investment problems.

While the original motivation comes from financial problems, the current input model has little to do with realistic financial markets, and is selected for its mathe-

---

matical cleanness and its relation to fundamental problems in stochastic games. The
main difference between our model and more realistic financial problems is that in
usual stock trading, optimizating rank-related quantities is not always correlated to
optimizing profits in the dollar amount. However, there are some strong similarities
as well, such as exotic financial derivatives based on quantities similar to ranks [20].

The current formulation is closely related to an important mathematical prob-
lem known as the *secretary problem* [11, 6], which has become a standard textbook
example [3, 5, 19], and has been the basis for many interesting extensions (includ-
ing [1, 14, 15, 17, 18]). The secretary problem comes from the following scenario:
A set of candidates for a single secretarial position are presented in random order.
The interviewer sees the candidates one at a time, and must make a decision to hire
or not to hire immediately upon seeing each candidate. Once a candidate is passed
over, the interviewer may not go back and hire that candidate. The general goal is to
maximize either the probability of selecting the top candidate, or the expected rank of
the selected candidate. This problem has also been stated with the slightly different
story of a princess selecting a suitor [3, p. 110]. More will be made of the relationship
between our current problem and the secretary problem in §2, and for further reading
on the secretary problem, we refer the reader to the survey by Freeman [10].

As mentioned above, we assume that the input comes from a random source in
which all permutations of final ranks $1, 2, \ldots, n$ are equally likely. Thus, each rank
x_i is uniformly distributed over the set $\{1, 2, \ldots, i\}$, and all ranks are independent of
one another. In fact, this closely parallels the most popular algorithm for generating
a random permutation [13, p. 139]. A natural question to ask is, knowing the relative
rank x_i of the current input, what is the expected final rank of this item (i.e., $E[r_i|x_i]$)?
Due to the uniform nature of the input source, the final rank of the ith item simply
scales up with the number of items left in the input sequence, and so $E[r_i|x_i] = \frac{n+1}{i+1}x_i$
(a simple proof of this is given in Appendix A).

Since all input ranks x_i are independent and uniformly distributed, little can
be inferred about the future inputs. We consider games in which a player watches
the stream of inputs, and can select items as they are seen; however, if an item is
passed up then it is gone for good and may not be selected later. We are interested
in strategies for two such games:

- Single pair selection: In this game, the player should make two selections, the
  first being the *low selection* and the second being the *high selection*. The goal
  of the player is to maximize the difference between the final ranks of these
  two selections. If the player picks the low selection upon seeing input x_ℓ at
  time step ℓ, and picks the high selection as input x_h at time step h, then the
  profit given to the player at the end of the game is the difference in final ranks
  of these items: $r_h - r_\ell$.
- Multiple pair selection: In this game, the player makes multiple choices of
  low/high pairs. At the end of the game the difference in final ranks of each
  selected pair of items is taken, and the differences for all pairs are added up
  to produce the player's final profit.

The strategies for these games share a common difficulty: If the player waits too long
to make the low selection, he risks not having enough choices for a good high selection;
however, making the low selection too early may result in an item selected before any
truly low items have been seen. The player in the second game can afford to be less
selective. If one chosen pair does not give a large difference, there may still be many
other pairs that are good enough to make up for this pair's small difference.

We present optimal solutions to both of the games. For the first game, where the player makes a single low selection and a single high selection, our strategy has expected profit $n - O(1)$. From the derivation of our strategy, it will be clear that the strategy is optimal. Even with full knowledge of the final ranks of all input items, the best expected profit in this game is less than n, and so in standard terms of on-line performance measurement [12, 16], the competitive ratio[1] of our strategy is one. The strength of our on-line strategy is rather intriguing.

For the second game, where multiple low/high pairs are selected, we provide an optimal strategy with expected profit $\frac{1}{8}n^2 - O(n \log n)$. For this problem, the optimal off-line strategy has expected profit of approximately $\frac{1}{6}n^2$, and so the competitive ratio of our strategy is $\frac{4}{3}$.

2. Single Low/High Selection. This section considers a scenario in which the player may pick a single item as the low selection, and a single later item as the high selection. If the low selection is made at time step ℓ and the high selection is made at time step h, then the expected profit is $E[r_h - r_\ell]$. The player's goal is to use a strategy for picking ℓ and h in order to maximize this expected profit.

As mentioned in the previous section, this problem is closely related to the secretary problem. A great deal of work has been done on the secretary problem and its variations, and this problem has taken a fundamental role in the study of games against a stochastic opponent. Our work extends the secretary problem, and gives complete solutions to two natural variants that have not previously appeared in the literature.

Much insight can be gained by looking at the optimal solution to the secretary problem, so we first sketch that solution below (using terminology from our problem about a "high selection"). To maximize the expected rank of a single high selection, we define the optimal strategy recursively using the following two functions:

$\mathcal{H}_n(i)$: This is a limit such that the player selects the current item if $x_i \geq \mathcal{H}_n(i)$.

$R_n(i)$: This is the expected final rank of the high selection if the optimal strategy is followed starting at the ith time step.

Since all permutations of the final ranks are equally likely, if the ith input item has rank x_i among the first i data items, then its expected final rank is $\frac{n+1}{i+1}x_i$. Thus, an optimal strategy for the secretary problem is to select the ith input item if and only if its expected final rank is better than could be obtained by passing over this item and using the optimal strategy from step $i + 1$ on. In other words, select the item at time step $i < n$ if and only if

$$\frac{n+1}{i+1}x_i \geq R_n(i+1).$$

If we have not made a selection before the nth step, then we must select the last item, whose rank is uniformly distributed over the range of integers from 1 to n — and so the expected final rank in that case is $R_n(n) = \frac{n+1}{2}$. For $i < n$ we can also define

$$\mathcal{H}_n(i) = \left\lceil \frac{i+1}{n+1} R_n(i+1) \right\rceil,$$

---

and to force selection at the last time step define $\mathcal{H}_n(n) = 0$. Furthermore, given this definition for $\mathcal{H}_n(i)$, the optimal strategy at step i depends only on the rank of the current item (which is uniformly distributed over the range $1, \ldots, i$) and the optimal strategy at time $i+1$. This allows us to recursively define $R_n(i)$ as follows when $i < n$:

$$
\begin{aligned}
R_n(i) &= \frac{\mathcal{H}_n(i) - 1}{i} R_n(i+1) + \sum_{j=\mathcal{H}_n(i)}^{i} \frac{1}{i} \cdot \frac{n+1}{i+1} j \\
&= \frac{\mathcal{H}_n(i) - 1}{i} R_n(i+1) + \frac{n+1}{i(i+1)} \cdot \frac{(i + \mathcal{H}_n(i))(i - \mathcal{H}_n(i) + 1)}{2} \\
&= \frac{\mathcal{H}_n(i) - 1}{i} \left(R_n(i+1) - \frac{n+1}{2(i+1)} \mathcal{H}_n(i) \right) + \frac{n+1}{2} .
\end{aligned}
$$

Since $\mathcal{H}_n(n) = 0$ and $R_n(n) = \frac{n+1}{2}$, we have a full recursive specification of both the optimal strategy and the performance of the optimal strategy. The performance of the optimal strategy, taken from the beginning, is $R_n(1)$. This value can be computed by the recursive equations, and was proved by Chow et al. to tend to $n + 1 - c$, for $c \approx 3.8695$, as $n \to \infty$ [6]. Furthermore, the performance approaches this limit from above, so for all n we have performance greater than $n - 2.87$.

For single pair selection, once a low selection is made we want to maximize the expected final rank of the high selection. If we made the low selection at step i, then we can optimally make the high selection by following the above strategy for the secretary problem, which results in an expected high selection rank of $R_n(i+1)$. How do we make the low selection? We can do this optimally by extending the recursive definitions given above with two new functions:

$\mathcal{L}_n(i)$:   This is a limit such that the player selects the current item if
          $x_i \le \mathcal{L}_n(i)$.
$P_n(i)$:   This is the expected high-low difference if the optimal strategy for
          making the low and high selections is followed starting at step i.

Thus, if we choose the ith input as the low selection, the expected profit is $R_n(i+1) - \frac{n+1}{i+1} x_i$. We should select this item if that expected profit is no less than the expected profit if we skip this item. This leads to the definition of $\mathcal{L}_n(i)$:

$$
\mathcal{L}_n(i) = \begin{cases} 0 & \text{if } i = n, \\ \left\lfloor \frac{i+1}{n+1} (R_n(i+1) - P_n(i+1)) \right\rfloor & \text{if } i < n. \end{cases}
$$

Using $\mathcal{L}_n(i)$, we derive the following profit function:

$$
P_n(i) = \begin{cases} 0 & \text{if } i = n, \\ P_n(i+1) + \frac{\mathcal{L}_n(i)}{i} \left(R_n(i+1) - P_n(i+1) - \frac{n+1}{i+1} \cdot \frac{\mathcal{L}_n(i)+1}{2} \right) & \text{if } i < n. \end{cases}
$$

From the derivation, it is clear that this is the optimal strategy, and can be implemented by using the recursive formulas to compute the $\mathcal{L}_n(i)$ values. The expected profit of our algorithm is given by $P_n(1)$, which is bounded in the following theorem.

THEOREM 2.1. *Our on-line algorithm for single low/high selection is optimal and has expected profit $n - O(1)$.*

Proof. It suffices to prove that a certain inferior algorithm has expected profit $n - O(1)$. The inferior algorithm is as follows: Use the solution to the secretary problem

to select, from the first $\lfloor n/2 \rfloor$ input items, an item with the minimum expected final rank. Similarly, pick an item with maximum expected rank from the second $\lceil n/2 \rceil$ inputs. For simplicity, we initially assume that n is even; see comments at the end of the proof for odd n. Let ℓ be the time step in which the low selection is made, and h the time step in which the high selection is made. Using the bounds from Chow et al. [6], we can bound the expected profit of this inferior algorithm by

$$
\begin{aligned}
E[r_h - r_\ell] &= E[r_h] - E[r_l] \geq \frac{n+1}{n/2+1}(n/2+1-c) - \frac{n+1}{n/2+1}c \\
&= \frac{n+1}{n+2}(n+2-4c) = n+1-4c+\frac{4c}{n+2} \; .
\end{aligned}
$$

Chow et al. [6] show that $c \leq 3.87$, and so the expected profit of the inferior algorithm is at least $n - 14.48$. For odd n, the derivation is almost identical, with only a change in the least significant term; specifically, the expected profit of the inferior algorithm for odd n is $n+1-4c+\frac{4c}{n+3}$, which again is at least $n - 14.48$. $\qquad\square$

3. Multiple Low/High Selection. This section considers a scenario in which the player again selects a low item followed by a high item, but may repeat this process as often as desired. If the player makes k low and high selections at time steps $\ell_1, \ell_2, \ldots, \ell_k$ and $h_1, h_2, \ldots, h_k$, respectively, then we require that

$$
1 \leq \ell_1 < h_1 < \ell_2 < h_2 < \cdots < \ell_k < h_k \leq n.
$$

The expected profit resulting from these selections is

$$
E[r_{h_1} - r_{\ell_1}] + E[r_{h_2} - r_{\ell_2}] + \cdots + E[r_{h_k} - r_{\ell_k}].
$$

3.1. Off-line Analysis. Let *interval j* refer to the time period between the instant of input item j arriving and the instant of input item $j+1$ arriving. For a particular sequence of low and high selections, we call interval j *active* if $\ell_i \leq j < h_i$ for some index i. We then amortize the total profit of a particular algorithm B by defining the amortized profit $A_B(j)$ for interval j to be

$$
A_B(j) = \begin{cases} r_{j+1} - r_j & \text{if interval } j \text{ is active,} \\ 0 & \text{otherwise.} \end{cases}
$$

Note that for a fixed sequence of low/high selections, the sum of all amortized profits is exactly the total profit, i.e.,

$$
\sum_{j=1}^{n} A_B(j) = \sum_{j=\ell_1}^{h_1-1}(r_{j+1}-r_j) + \sum_{j=\ell_2}^{h_2-1}(r_{j+1}-r_j) + \cdots + \sum_{j=\ell_k}^{h_k-1}(r_{j+1}-r_j)
$$

$$
= (r_{h_1} - r_{\ell_1}) + (r_{h_2} - r_{\ell_2}) + \cdots + (r_{h_k} - r_{\ell_k}).
$$

For an off-line algorithm to maximize the total profit we need to maximize the amortized profit, which is done for a particular sequence of r_i's by making interval j active if and only if $r_{j+1} > r_j$. Translating this back to the original problem of making low and high selections, this is equivalent to identifying all maximal-length increasing intervals and selecting the beginning and ending points of these intervals as low and high selections, respectively. These observations and some analysis give the following lemma.

LEMMA 3.1. *The optimal off-line algorithm just described has expected profit* $\frac{1}{6}\left(n^2 - 1\right)$.

Proof. This analysis is performed by examining the expected amortized profits for individual intervals. In particular, for any interval j,

$$
\begin{aligned}
E[A_{OFF}(j)] &= Pr[r_{j+1} > r_j] \cdot E[A_j | r_{j+1} > r_j] + Pr[r_{j+1} < r_j] \cdot E[A_j | r_{j+1} < r_j] \\
&= \frac{1}{2} \cdot E[r_{j+1} - r_j | r_{j+1} > r_j] + \frac{1}{2} \cdot 0 \\
&= \frac{1}{2} \sum_{i=1}^{n-1} \sum_{k=i+1}^{n} \frac{Pr[r_{j+1} = k \text{ and } r_j = i]}{Pr[r_{j+1} > r_j]} \cdot (k - i) \\
&= \frac{1}{2} \sum_{i=1}^{n-1} \sum_{k=i+1}^{n} \frac{2}{n(n-1)} (k - i) \\
&= \frac{1}{2} \cdot \frac{2}{n(n-1)} \cdot \frac{(n+1)n(n-1)}{6} \\
&= \frac{n+1}{6}.
\end{aligned}
$$

Since there are $n - 1$ intervals and the above analysis is independent of the interval number j, summing the amortized profit over all intervals gives the expected profit stated in the lemma. □

3.2. On-line Analysis. In our on-line algorithm for multiple pair selection, there are two possible states: FREE and HOLDING. In the FREE state, we choose the current item as a low selection if $x_i < \frac{i+1}{2}$; furthermore, if we select an item then we move from the FREE state into the HOLDING state. On the other hand, in the HOLDING state if the current item has $x_i > \frac{i+1}{2}$, then we choose this item as a high selection and move into the FREE state. We name this algorithm OP, which can stand for "opportunistic" since this algorithm makes a low selection whenever the probability is greater than $\frac{1}{2}$ that the next input item will be greater than this one. Later we will see that the name OP could just as well stand for "optimal," since this algorithm is indeed optimal.

The following lemma gives the expected profit of this algorithm. In the proof of this lemma we use the following equality:

$$
\sum_{i=1}^{k} \frac{2i}{2i+1} = k + 1 + \frac{1}{2} H_k - H_{2k+1}.
$$

LEMMA 3.2. *The expected profit from our on-line algorithm is*

$$
E[P_{OP}] = \begin{cases} \dfrac{n+1}{8}\left(n + H_{\frac{n-2}{2}} - 2H_{n-1}\right) & \text{if n is even,} \\[2ex] \dfrac{n+1}{8}\left(n + H_{\frac{n-1}{2}} - 2H_n + \dfrac{1}{n}\right) & \text{if n is odd.} \end{cases}
$$

In cleaner forms we have $E[P_{OP}] = \frac{n+1}{8}(n - H_n + \Theta(1)) = \frac{1}{8}n^2 - \Theta(n \log n)$.

Proof. Let R_i be the random variable of the final rank of the ith input item. Let $A_{OP}(i)$ be the amortized cost for interval i as defined in §3.1. Since $A_{OP}(i)$ is nonzero

only when interval i is active,

$$
\begin{aligned}
E[A_{OP}(i)] &= E[A_{OP}(i)|\text{Interval } i \text{ is active}] \cdot Prob[\text{Interval } i \text{ is active}] \\
&= E[R_{i+1} - R_i|\text{Interval } i \text{ is active}] \cdot Prob[\text{Interval } i \text{ is active}].
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
E[P_{OP}] &= \sum_{i=1}^{n-1} E[A_{OP}(i)] \\
&= \sum_{i=1}^{n-1} E[R_{i+1} - R_i|\text{Interval } i \text{ is active}] \cdot Prob[\text{Interval } i \text{ is active}].
\end{aligned}
$$

Under what conditions is an interval active? If $x_i < \frac{i+1}{2}$ this interval is certainly active. If the algorithm was not in the HOLDING state prior to this step, it would be after seeing input x_i. Similarly, if $x_i > \frac{i+1}{2}$ the algorithm must be in the FREE state during this interval, and so the interval is not active. Finally, if $x_i = \frac{i+1}{2}$ the state remains what it has been for interval $i - 1$. Furthermore, since i must be odd for this case to be possible, $i - 1$ is even, and x_{i-1} cannot be $\frac{i}{2}$ (and thus x_{i-1} unambiguously indicates whether interval i is active). In summary, determining whether interval i is active requires looking at only x_i and occasionally x_{i-1}. Since the expected amortized profit of step i depends on whether i is odd or even, we break the analysis up into these two cases below.

Case 1: i is even. Note that $Prob[x_i < \frac{i+1}{2}] = \frac{1}{2}$, and x_i cannot be exactly $\frac{i+1}{2}$, which means that with probability $\frac{1}{2}$ interval i is active. Furthermore, R_{i+1} is independent of whether interval i is active or not, and so

$$
\begin{aligned}
E[A_{OP}(i)|\text{Interval } i \text{ is active}] &= E[R_{i+1}] - E[R_i|\text{Interval } i \text{ is active}] \\
&= \frac{n+1}{2} - \frac{n+1}{i+1} \sum_{j=1}^{i/2} \frac{2}{i} j \\
&= \frac{n+1}{2} - \frac{n+1}{i+1} \cdot \frac{2}{i} \cdot \frac{i(i+2)}{8} \\
&= \frac{n+1}{4} \cdot \frac{i}{i+1}.
\end{aligned}
$$

Case 2: i is odd. Since interval 1 cannot be active, we assume that $i \geq 3$. We need to consider the case in which $x_i = \frac{i+1}{2}$, and so

$$
\begin{aligned}
&Prob[\text{Interval } i \text{ is active}] \\
&= Prob[x_i < \frac{i+1}{2}] + Prob[x_i = \frac{i+1}{2}] \cdot Prob[x_{i-1} < \frac{i}{2}] \\
&= \frac{i-1}{2i} + \frac{1}{i} \cdot \frac{1}{2} = \frac{1}{2}.
\end{aligned}
$$

Computing the expected amortized cost of interval i is slightly more complex than in Case 1.

$$
\begin{aligned}
&E[A_{OP}(i)|\text{Interval } i \text{ is active}] \\
&= E[R_{i+1}] - E[R_i|\text{Interval } i \text{ is active}]
\end{aligned}
$$

$$= \quad \frac{n+1}{2} - \frac{n+1}{i+1} \left(\sum_{j=1}^{(i-1)/2} \frac{2}{i} j + \frac{1}{i} \cdot \frac{i+1}{2} \right)$$

$$= \quad \frac{n+1}{2} - \frac{n+1}{i+1} \left(\frac{2}{i} \cdot \frac{(i-1)(i+1)}{8} + \frac{1}{i} \cdot \frac{i+1}{2} \right)$$

$$= \quad \frac{n+1}{2} - \frac{n+1}{i+1} \cdot \frac{(i+1)(i+1)}{4i}$$

$$= \quad \frac{n+1}{4} \cdot \frac{i-1}{i}.$$

Combining both cases,

$$E[P_{OP}] \quad = \quad \sum_{i=1}^{n-1} E[A_{OP}(i) | \text{Interval } i \text{ is active}] \cdot Prob[\text{Interval } i \text{ is active}]$$

$$= \quad \frac{n+1}{8} \left(\sum_{k=1}^{\lfloor (n-2)/2 \rfloor} \frac{2k}{2k+1} + \sum_{k=1}^{\lfloor (n-1)/2 \rfloor} \frac{2k}{2k+1} \right),$$

where the first sum accounts for the odd terms of the original sum, and the second sum accounts for the even terms.

When n is even this sum becomes

$$E[P_{OP}] \quad = \quad \frac{n+1}{8} \left(\sum_{k=1}^{\lfloor (n-2)/2 \rfloor} \frac{2k}{2k+1} + \sum_{k=1}^{\lfloor (n-1)/2 \rfloor} \frac{2k}{2k+1} \right)$$

$$= \quad \frac{n+1}{8} \left(2 \sum_{k=1}^{(n-2)/2} \frac{2k}{2k+1} \right)$$

$$= \quad \frac{n+1}{8} \left(n + H_{\frac{n-2}{2}} - 2H_{n-1} \right),$$

which agrees with the claim in the lemma. When n is odd the sum can be simplified as

$$E[P_{OP}] \quad = \quad \frac{n+1}{8} \left(\sum_{k=1}^{\lfloor (n-2)/2 \rfloor} \frac{2k}{2k+1} + \sum_{k=1}^{\lfloor (n-1)/2 \rfloor} \frac{2k}{2k+1} \right)$$

$$= \quad \frac{n+1}{8} \left(2 \sum_{k=1}^{(n-1)/2} \frac{2k}{2k+1} - \frac{n-1}{n} \right)$$

$$= \quad \frac{n+1}{8} \left(n + H_{\frac{n-1}{2}} - 2H_n + \frac{1}{n} \right),$$

which again agrees with the claim in the lemma. The simplified forms follow the fact that for any odd $n \geq 3$ we can bound $\frac{1}{n} \leq H_n - H_{\frac{n-1}{2}} \leq \ln 2 + \frac{1}{n}$.   □

Combining this result with that of §3.1, we see that our on-line algorithm has expected profit 3/4 of what could be obtained with full knowledge of the future. In terms of competitive analysis, our algorithm has competitive ratio 4/3, which means that not knowing the future is not terribly harmful in this problem!

3.3. Optimality of Our On-Line Algorithm. This section proves that algorithm OP is optimal. We will denote permutations by a small Greek letter with a subscript giving the size of the permutation; in other words, a permutation on the set $\{1, 2, \ldots, i\}$ may be denoted ρ_i or σ_i.

A permutation on i items describes fully the first i inputs to our problem, and given such a permutation we can also compute the permutation described by the first $i-1$ inputs (or $i-2$, etc.). We will use the notation $\sigma_i|_{i-1}$ to denote such a restriction. This is not just a restriction of the domain of the permutation to $\{1, \ldots, i-1\}$, since unless $\sigma_i(i) = i$ this simplistic restriction will not form a valid permutation.

Upon seeing the ith input, an algorithm may make one of the following moves: it may make this input a low selection; it may make this input a high selection; or it may simply ignore the input and wait for the next input. Therefore, any algorithm can be entirely described by a function which maps permutations (representing inputs of arbitrary length) into this set of moves. We denote such a move function for algorithm B by M_B, which for any permutation σ_i maps $M_B(\sigma_i)$ to an element of the set $\{$ "low", "high", "wait"$\}$. Notice that not all move functions give valid algorithms. For example, it is possible to define a move function that makes two low selections in a row for certain inputs, even though this is not allowed by our problem.

We define a generic HOLDING state just as we did for our algorithm. An algorithm is in the HOLDING state at time i if it has made a low selection, but has not yet made a corresponding high selection. For algorithm B we define the set $L_B(i)$ to be the set of permutations on i items that result in the algorithm being in the HOLDING state after processing these i inputs. We explicitly define these sets using the move function:

$$L_B(i) = \begin{cases} \{\sigma_i | M_B(\sigma_i) = \text{``low''}\} & \text{if } i = 1, \\ \{\sigma_i | M_B(\sigma_i) = \text{``low''} \text{ or} \\ \quad (M_B(\sigma_i) = \text{``wait''} \text{ and } \sigma_i|_{i-1} \in L_B(i-1))\} & \text{if } i > 1. \end{cases}$$

The $L_B(i)$ sets are all we need to compute the expected amortized profit for interval i, since

$$\begin{aligned} E[A_B(i)] &= Prob[\text{Interval } i \text{ is active}] \cdot E[R_{i+1} - R_i | \text{Interval } i \text{ is active}] \\ &= \frac{|L_B(i)|}{i!} \left(\frac{n+1}{2} - \frac{n+1}{i+1} \sum_{\rho_i \in L_B(i)} \frac{1}{|L_B(i)|} \rho_i(i) \right) \\ &= \frac{n+1}{i!} \left(\frac{|L_B(i)|}{2} - \frac{1}{i+1} \sum_{\rho_i \in L_B(i)} \rho_i(i) \right). \end{aligned}$$

We use the above notation and observations to prove the optimality of algorithm OP.

THEOREM 3.3. *Algorithm OP is an optimal algorithm for the multiple pair selection problem.*

Proof. Since the move functions (which define specific algorithms) work on permutations, we will fix an ordering of permutations in order to compare strategies. We order permutations first by their size, and then by a lexicographic ordering of the actual permutations. When comparing two different algorithms B and C, we start enumerating permutations in this order and count how many permutations cause the same move in B and C, stopping at the first permutation σ_i for which $M_B(\sigma_i) \neq M_C(\sigma_i)$, i.e., the first permutation for which the algorithms make different moves. We call the

number of permutations that produce identical moves in this comparison process the *length of agreement between B and C*.

To prove the optimality of our algorithm by contradiction, we assume that it is not optimal, and of all the optimal algorithms let B be the algorithm with the longest possible length of agreement with our algorithm OP. Let σ_k be the first permutation in which $M_B(\sigma_k) \neq M_{OP}(\sigma_k)$. Since B is different from OP at this point, at least one of the following cases must hold:

(a) $\sigma_k|_{k-1} \notin L_B(k-1)$ and $\sigma_k(k) < \frac{k+1}{2}$ and $M_B(\sigma_k) \neq$ "low" (i.e., algorithm B is not in the HOLDING state, gets a low rank input, but does not make it a low selection).

(b) $\sigma_k|_{k-1} \notin L_B(k-1)$ and $\sigma_k(k) \geq \frac{k+1}{2}$ and $M_B(\sigma_k) \neq$ "wait" (i.e., algorithm B is not in the HOLDING state, gets a high rank input, but makes it a low selection anyway).

(c) $\sigma_k|_{k-1} \in L_B(k-1)$ and $\sigma_k(k) > \frac{k+1}{2}$ and $M_B(\sigma_k) \neq$ "high" (i.e., algorithm B is in the HOLDING state, gets a high rank input, but doesn't make it a high selection).

(d) $\sigma_k|_{k-1} \in L_B(k-1)$ and $\sigma_k(k) \leq \frac{k+1}{2}$ and $M_B(\sigma_k) \neq$ "wait" (i.e., algorithm B is in the HOLDING state, gets a low rank input, but makes it a high selection anyway).

In each case, we will show how to transform algorithm B into a new algorithm C such that C performs at least as well as B, and the length of agreement between C and OP is longer than that between B and OP. This provides the contradiction that we need.

Case (a): Algorithm C's move function is identical to B's except for the following values:

$$M_C(\sigma_k) = \text{"low"},$$

$$M_C(\rho_{k+1}) = \begin{cases} \text{"high"} & \text{if } \rho_{k+1}|_k = \sigma_k \text{ and } M_B(\sigma_{k+1}) = \text{"wait"} , \\ \text{"wait"} & \text{if } \rho_{k+1}|_k = \sigma_k \text{ and } M_B(\sigma_{k+1}) = \text{"low"} , \\ M_B(\rho_{k+1}) & \text{otherwise.} \end{cases}$$

In other words, algorithm C is the same as algorithm B except that we "correct B's error" of not having made this input a low selection. The changes of the moves on input $k+1$ insures that $L_C(k+1)$ is the same as $L_B(k+1)$. It is easily verified that the new sets $L_C(i)$ (corresponding to the HOLDING state) are identical to the sets $L_B(i)$ for all $i \neq k$. The only difference at k is the insertion of σ_k, i.e., $L_C(k) = L_B(k) \cup \{\sigma_k\}$.

Let P_B and P_C be the profits of B and C, respectively. Since their amortized costs differ only at interval k,

$$\begin{aligned} E[P_C &- P_B] \\ &= E[A_C(k)] - E[A_B(k)] \\ &= \frac{n+1}{k!} \left(\frac{|L_C(k)|}{2} - \frac{1}{k+1} \sum_{\rho_k \in L_C(k)} \rho_k(k) \right) \\ &\quad - \frac{n+1}{k!} \left(\frac{|L_B(k)|}{2} - \frac{1}{k+1} \sum_{\rho_k \in L_B(k)} \rho_k(k) \right) \\ &= \frac{n+1}{k!} \left(\frac{1}{2} - \frac{1}{k+1} \sigma_k(k) \right) . \end{aligned}$$

By one of the conditions of Case (a), $\sigma_k(k) < \frac{k+1}{2}$, so we finish this derivation by noting that

$$E[P_C - P_B] = \frac{n+1}{k!}\left(\frac{1}{2} - \frac{1}{k+1}\sigma_k(k)\right) > \frac{n+1}{k!}\left(\frac{1}{2} - \frac{1}{k+1} \cdot \frac{k+1}{2}\right) = 0.$$

Therefore, the expected profit of algorithm C is greater than that of B.

Case (b): As in Case (a) we select a move function for algorithm C that causes only one change in the sets of HOLDING states, having algorithm C not make input k a low selection. In particular, these sets are identical with those of algorithm B with the one exception that $L_C(k) = L_B(k) - \{\sigma_k\}$. Analysis similar to Case (a) shows

$$E[P_C - P_B] = \frac{n+1}{k!}\left(\frac{1}{k+1}\sigma_k(k) - \frac{1}{2}\right) \geq \frac{n+1}{k!}\left(\frac{1}{k+1} \cdot \frac{k+1}{2} - \frac{1}{2}\right) = 0.$$

Case (c): In this case we select a move function for algorithm C such that $L_C(k) = L_B(k) - \{\sigma_k\}$, resulting in algorithm C selecting input k as a high selection, and giving an expected profit gain of

$$E[P_C - P_B] = \frac{n+1}{k!}\left(\frac{1}{k+1}\sigma_k(k) - \frac{1}{2}\right) > \frac{n+1}{k!}\left(\frac{1}{k+1} \cdot \frac{k+1}{2} - \frac{1}{2}\right) = 0.$$

Case (d): In this case we select a move function for algorithm C such that $L_C(k) = L_B(k) \cup \{\sigma_k\}$, resulting in algorithm C not taking input k as a high selection, and giving an expected profit gain of

$$E[P_C - P_B] = \frac{n+1}{k!}\left(\frac{1}{2} - \frac{1}{k+1}\sigma_k(k)\right) \geq \frac{n+1}{k!}\left(\frac{1}{2} - \frac{1}{k+1} \cdot \frac{k+1}{2}\right) = 0.$$

In each case, we transformed algorithm B into a new algorithm C that performs at least as well (and hence must be optimal), and has a longer length of agreement with algorithm OP than B does. This directly contradicts our selection of B as the optimal algorithm with the longest length of agreement with OP, and this contradiction finishes the proof that algorithm OP is optimal.   □

4. Conclusion. In this paper, we examined a natural on-line problem related to both financial games and the classic secretary problem. We select low and high values from a randomly ordered set of values presented in an on-line fashion, with the goal of maximizing the difference in final ranks of such low/high pairs. We considered two variations of this problem. The first allowed us to choose only a single low value followed by a single high value from a sequence of n values, while the second allowed selection of arbitrarily many low/high pairs. We presented provably optimal algorithms for both variants, gave tight analyses of the performance of these algorithms, and analyzed how well the on-line performance compares to the optimal off-line performance.

Our paper opens up many problems. Two particularly interesting directions are to consider more realistic input sources and to maximize quantities other than the difference in rank.

Appendix. Proof of Expected Final Rank. In this appendix section, we prove that if an item has relative rank x_i among the first i inputs, then its expected rank r_i among all n inputs is given by $E[r_i|x_i] = \frac{n+1}{i+1}x_i$.

LEMMA A.1. *If a given item has rank x from among the first i inputs, and if the $i + 1$st input is uniformly distributed over all possible rankings, then the expected rank of the given item among the first $i + 1$ inputs is $\frac{i+2}{i+1}x$.*

Proof. If we let R be a random variable denoting the rank of our given item from among the first $i + 1$ inputs, then we see that the value of R depends on the rank of the $i + 1$st input. In particular, if the rank of the $i + 1$st input is $\leq x$ (which happens with probability $\frac{x}{i+1}$), then the new rank of our given item will be $x + 1$. On the other hand, if the rank of the $i + 1$st input is $> x$ (which happens with probability $\frac{i+1-x}{i+1}$), then the rank of our given item is still x among the first $i + 1$ inputs. Using this observation, we see that

$$E[R] = \frac{x}{i+1}(x+1) + \frac{i+1-x}{i+1}x = \frac{x+1+i+1-x}{i+1}x = \frac{i+2}{i+1}x,$$

which is what is claimed in the lemma.   □

For a fixed position i, the above extension of rank to position $i + 1$ is a constant times the rank of the item among the first i inputs. Because of this, we can simply extend this lemma to the case where x is not a fixed rank but is a random variable, and we know the expected rank among the first i items.

COROLLARY A.2. *If a given item has expected rank x from among the first i inputs, and if the $i + 1$st input is uniformly distributed over all possible rankings, then the expected rank of the given item among the first $i + 1$ inputs is $\frac{i+2}{i+1}x$.*

Simply multiplying together the change in expected rank from among i inputs, to among $i + 1$ inputs, to among $i + 2$ inputs, and so on up to n inputs, we get a telescoping product with cancellations between successive terms, resulting in the following corollary.

COROLLARY A.3. *If a given item has rank x from among the first i inputs, and if the remaining inputs are uniformly distributed over all possible rankings, then the expected rank of the given item among all n inputs is $\frac{n+1}{i+1}x$.*

REFERENCES

[1] M. AJTAI, N. MEGIDDO, AND O. WAARTS, *Improved algorithms and analysis for secretary problems and generalizations*, in Proceedings of the 36th Symposium on Foundations of Computer Science, 1995, pp. 473–482.

[2] G. J. ALEXANDER AND W. F. SHARPE, *Fundamentals of Investments*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[3] P. BILLINGSLEY, *Probability and Measure*, John Wiley and Sons, New York, second ed., 1986.

[4] A. CHOU, J. COOPERSTOCK, R. EL-YANIV, M. KLUGERMAN, AND T. LEIGHTON, *The statistical adversary allows optimal money-making trading strategies*, in Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, 1995, pp. 467–476.

[5] Y. CHOW, H. ROBBINS, AND D. SIEGMUND, *Great Expectations: The Theory of Optimal Stopping*, Houghton Mifflin Co., Boston, 1971.

[6] Y. S. CHOW, S. MORIGUTI, H. ROBBINS, AND S. M. SAMUELS, *Optimal selection based on relative rank (the "secretary problem")*, Israel J. Math., 2 (1964), pp. 81–90.

[7] T. M. COVER, *An algorithm for maximizing expected log investment return*, IEEE Transactions on Information Theory, IT-30 (1984), pp. 369–373.

[8] R. EL-YANIV, A. FIAT, R. KARP, AND G. TURPIN, *Competitive analysis of financial games*, in Proceedings of the 33rd Symposium on Foundations of Computer Science, 1992, pp. 327–333.

[9] E. F. FAMA, *Foundations of Finance*, Basic Books, New York, NY, 1976.

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

[11] M. GARDNER, *Mathematical games*, Scientific American, 202 (1960), pp. 150–153.

[12] A. R. KARLIN, M. S. MANASSE, L. RUDOLPH, AND D. D. SLEATOR, *Competitive snoopy caching*, in Proceedings of the 27th Symposium on Foundations of Computer Science, 1986, pp. 244–254.

[13] D. E. KNUTH, *The Art of Computer Programming; Volume 2, Seminumerical Algorithms*, Addison-Wesley Publishing Co., Reading, MA, second ed., 1981.

[14] A. MUCCI, *Differential equations and optimal choice problems*, Ann. Statist., 1 (1973), pp. 104–113.

[15] W. T. RASMUSSEN AND S. R. PLISKA, *Choosing the maximum from a sequence with a discount function*, Appl. Math. Optimization, 2 (1976), pp. 279–289.

[16] D. D. SLEATOR AND R. E. TARJAN, *Amortized efficiency of list update and paging rules*, Comm. ACM, 28 (1985), pp. 202–208.

[17] M. H. SMITH AND J. J. DEELY, *A secretary problem with finite memory*, J. Am. Statist. Assoc., 70 (1975), pp. 357–361.

[18] M. TAMAKI, *Recognizing both the maximum and the second maximum of a sequence*, J. Appl. Prob., 16 (1979), pp. 803–812.

[19] P. WHITTLE, *Optimization Over Time: Dynamic Programming and Stochastic Control; Volume 1*, John Wiley and Sons, Chichester, 1982.

[20] P. WILMOTT, S. HOWISON, AND J. DEWYNNE, *The Mathematics of Financial Derivatives*, Cambridge University Press, Cambridge, United Kingdom, 1995.