Hiring a Secretary from a Poset

Ravi Kumar
Yahoo! Research
701 First Ave
Sunnyvale, CA 94089.
ravikumar@yahoo-inc.com

Silvio Lattanzi*
Google Inc.
76 Ninth Avenue
New York, NY, 10011.
silviol@google.com

Sergei Vassilvitskii
Yahoo! Research
111 W 40th Street
New York, NY 10018.
sergei@yahoo-inc.com

Andrea Vattani*
UC San Diego
9500 Gilman Drive
La Jolla, CA 92093.
avattani@cs.ucsd.edu

ABSTRACT

The secretary problem lies at the core of mechanism design for online auctions. In this work we study the generalization of the classical secretary problem in a setting where there is only a *partial* order between the elements and the goal of the algorithm is to return one of the maximal elements of the poset. This is equivalent to the auction setting where the seller has a multidimensional objective function with only a partial order among the outcomes. We obtain an algorithm that succeeds with probability at least $k^{-k/(k-1)}\left(\left(1+\log k^{1/(k-1)}\right)^k-1\right)$, where k is the number of maximal elements in the poset and is the only information about the poset that is known to the algorithm; the success probability approaches the classical bound of $1/e$ as $k \to 1$. On the other hand, we prove an almost matching upper bound of $k^{-1/(k-1)}$ on the success probability of any algorithm for this problem; this upper bound holds even if the algorithm knows the complete structure of the poset.

Categories and Subject Descriptors

F.2.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*Nonumerical Algorithms and Problems*

General Terms

Algorithms, Theory

Keywords

Secretary problem, Partial order

*Part of this work was done while the authors were visiting Yahoo! Research.

1. INTRODUCTION

The secretary problem [7, 8] is a perfect example of online decision-making under uncertainty. The setting is humble: candidates for a secretary position arrive online in a random order and the goal is to choose the best candidate, with the constraint that no past decision can be reverted. The optimal algorithm is to skip the first $1/e$ fraction of the candidates and to choose the next arriving candidate who is the best seen so far; this algorithm yields a success probability of $1/e$. The secretary problem has a rich history dating back at least a century, and is a frequent object of study even to this day. See the survey article by Ferguson [8] for an excellent historical perspective of the secretary problem.

Implicit in the classical setting is the assumption that there is a total order on the candidates, but this assumption rarely holds in real life since candidates often have incomparable attributes. This leads to the natural *poset secretary problem*: if the elements of the permutation (candidates) are only partially ordered, how to maximize the probability of returning a maximal element in the poset? Note that the incomparable elements present the main challenge: many simple modifications of the total order algorithm to handle the incomparable elements can be shown to have vanishing success probabilities.

Secretary problems have recently been shown to lie at the core of online auction and mechanism design problems [4]. For instance, Hajiaghayi, Kleinberg, and Parkes [13] showed how to convert the classic secretary problem into a group strategy-proof mechanism for the online single item auction. The algorithm we present can be adapted in a similar fashion to a setting where the seller has a multidimensional utility function that does not lead to a total ordering on the bidders. The bidders arrive with potentially incomparable bids and the goal is to sell the item at a no-regret price, i.e., to select a bidder who is not dominated by any of the others.

The poset secretary problem was first studied by Preater [19], who proposed an algorithm with a success probability of 1/8. Recently, Georgiou et al. [10] improved this bound to 1/4; they also showed that this bound is tight for Preater's algorithm. These algorithms suffer from two major drawbacks. First, the success probability does not match the classical bound when the poset is a total order. Second, these bounds do not improve with the number of maximal elements in the poset, which is undesirable since the prob-

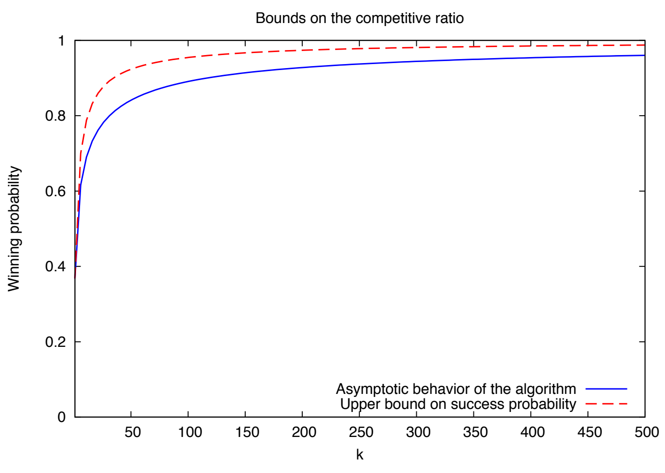

Figure 1: A visualization of the upper and lower bounds for the poset secretary problem as a function of the number of maximal elements in the poset.

lem should only become easier as the number of solutions grows.

1.1 Main results

In this paper we study the secretary problem in the partial order setting. We assume that we know k, the number of maximal elements in the poset. Our algorithms take on the standard form with one subtle difference. As before, we examine all of the elements up to a threshold and then consider the first undominated element. We select this element only if the poset at the time has at most k maximal elements. The latter condition may make us pass on a maximal element early in the sequence, but we will never pass on the last maximal element. We show that for a judicious choice of the threshold (that depends on k) our algorithm succeeds with probability roughly $k^{-\frac{k}{k-1}}\left(\left(1+\log k^{1/(k-1)}\right)^k-1\right)$; see Theorem 3 for a precise statement. This quantity recovers the $^1/_e$ bound in the limit as $k \to 1$, but quickly surpasses it, reaching 0.47 at $k=2$ and 0.52 at $k=3$. We show an almost matching upper bound of $k^{-1/(k-1)}+o(1)$ (Theorem 9), showing that no algorithm succeeds with probability better than 0.5 for $k=2$ and better than 0.57 for $k=3$. Figure 1 shows these bounds. Closing the gap between the two remains an interesting open problem.

On the technical side, we proceed as follows. To analyze the algorithm, we concentrate on the probability of the algorithm reaching the last maximal element. We introduce the concept of a *blocking set* and show that if the blocking set occurs early in a random permutation of elements, then the second condition on accepting an element (requiring that the number of maximal elements in the induced poset be at most k) prevents the algorithm from returning a suboptimal element. We then proceed to construct such permutations by starting from the end and increasing the suffix so as to keep the blocking sets early in the sequence.

For an upper bound on the success probability of any algorithm, we use a linear programming approach introduced in the work of Buchbinder, Jain, and Singh [6]. We use this approach to consider the very specific poset of k disjoint total orders, each on $^n/_k$ elements, and show that no algorithm has a good competitive ratio on this poset. To that end, we construct a linear program whose value upper bounds the probability of success of any algorithm on this poset. We then present a feasible solution to the corresponding dual, thus establishing a bound on the success probability of any algorithm for this problem. It is worth to note that this bound holds even if the algorithm knows that the poset under consideration consists of k disjoint total orders.

1.2 Related work

Independently and concurrently with our work, Freij and Wästlund [9] recently proposed an algorithm for the partially ordered secretary problem and claimed a bound of $^1/_e$ on its success probability. Their algorithm works as follows. Assign a score uniformly at random in $[0,1]$ as elements arrive, skip the first $^1/_e$ fraction of elements, and pick the first element which is the greedy maximum of the poset seen so far including this element. Here, the greedy maximum of a poset with weighted elements is defined inductively: it is the lowest weight element if it is the maximal or it is the greedy maximum of the sub-poset induced by the elements that are bigger than the lowest weight element. Their algorithm has the same downside as that of [10, 19], namely, the competitive ratio does not increase with the number of maximal elements and remains bounded by $^1/_e$.

The poset secretary problem has been previously considered for specific cases of posets. For example, Morayne [18] and Kubicki et al. [16] present an optimal stopping time for the case of the complete binary tree. Gnedin [11] explored other specific poset structures. Generally, Bruss [5] defined a way (known as the Odds algorithm) to compute optimal stopping rules for any last-success problem, which applies to the classical version of the secretary problem as well as to the case of any known poset with a single maximal element. For poset-oblivious algorithms, very recently, Kozik [15] proposed a dynamic threshold algorithm that selects a maximal element of any poset with probability at least $1/4+\epsilon$ (for some small $\epsilon > 0$), therefore beating the $1/4$ bound of Georgiou et al. [10].

Other variants of secretary problems have been previously applied to the online auction setting. For example, designing mechanisms to maximize some function of the top k elements [1, 2, 14] or some function on the accepted set of elements, such as online submodular function maximization [12, 17], finding the heaviest weight independent set of a matroid [3, 20], etc. See [4] for a survey of some of these results.

2. PRELIMINARIES

Let U be a universe of n elements. A *poset* $\mathcal{P} \subseteq U^2$ is a binary relation that is reflexive, anti-symmetric, and transitive. We use $a \prec_{\mathcal{P}} b$ to denote $(a,b) \in \mathcal{P}$ and use $a \|_{\mathcal{P}} b$ to denote $(a,b) \notin \mathcal{P} \wedge (b,a) \notin \mathcal{P}$, i.e., a and b are incomparable. A *linear extension* of $\mathcal{P}$ is a permutation π on U such that $a \prec_{\mathcal{P}} b \implies \pi^{-1}(a) < \pi^{-1}(b)$.

An element a is *maximal* (aka a *secretary*) if there is no element b such that $a \prec_{\mathcal{P}} b$. Let $\max \mathcal{P}$ be the set of all secretaries of $\mathcal{P}$ and let $k = |\max \mathcal{P}|$, the number of secretaries.

We denote by $S_i = \bigcup_{j<i}\{\pi(j)\}$ the set of elements preceding i in the permutation.

Given $S \subseteq U$, let $\mathcal{P}|_S = \mathcal{P} \cap S^2$, the poset obtained from $\mathcal{P}$ by using the elements only in S.

DEFINITION 1 (PARETO FRONTIER). *Given a poset $\mathcal{P}$ and a subset $S \subseteq U$, the* Pareto frontier *$\mathcal{F}_{\mathcal{P}}(S)$ is defined to be $\max \mathcal{P}|_S$.*

3. ALGORITHM AND ANALYSIS

Let $\mathcal{P}$ be the given poset. We assume that the algorithm is given $k = |\max \mathcal{P}|$, the number of secretaries. Our algorithm proceeds in a way similar to the algorithm in the total order setting. It examines all of the elements before a threshold τ. An element a arriving after the threshold is returned if two conditions are met. First, the element must be *undominated*, i.e., $a \in \mathcal{F}_{\mathcal{P}}(S)$, where S is the set of all of elements seen thus far; since any dominated element cannot be maximal, this is without loss of generality. Second, the total size of $\mathcal{F}_{\mathcal{P}}(S)$ is at most k. While the second condition may lead the algorithm to pass on a maximal element, it will never pass on the last maximal element in the permutation.

We will denote by π the order in which the elements arrive. To describe the algorithm, let τ_k be the *stopping threshold*:

$$\tau_k = \begin{cases} n/e, & k = 1, \\ n/k^{\frac{1}{k-1}}, & k > 1. \end{cases}$$

Note that $\lim_{k \to 1^+} \tau_k = n/e$.

Algorithm 1 SECRETARY (π, k).

```
1: $S = \{\pi(1), \ldots, \pi(\tau_k)\}$
2: for $i = \tau_k + 1 \ldots n$ do
3:     $a = \pi(i)$
4:     $S \leftarrow S \cup \{a\}$
5:     if $a \in \mathcal{F}_{\mathcal{P}}(S) \wedge |\mathcal{F}_{\mathcal{P}}(S)| \le k$ then
6:         return $a$
7:     end if
8: end for
```

3.1 Warmup: Analysis for $k = 1$

THEOREM 2. *For any poset $\mathcal{P}$ with $k = 1$, Algorithm 1 succeeds with probability at least $1/e$.*

PROOF. Let $\mathcal{P}_L$ be an arbitrary linear extension of $\mathcal{P}$; by definition, if $a \prec_{\mathcal{P}_L} b$, then either $a \prec_{\mathcal{P}} b$ or $a \|_{\mathcal{P}} b$. Now, we compare the performance of Algorithm 1 on $\mathcal{P}$ and $\mathcal{P}_L$. Consider any permutation π such that the algorithm outputs the secretary when run on $\mathcal{P}_L$. We claim that the algorithm outputs the secretary when run on $\mathcal{P}$ as well. This will complete the proof since Algorithm 1 on a linear order (i.e., $\mathcal{P}_L$) is the optimal algorithm for the classical secretary problem, and therefore succeeds with probability at least $1/e$.

To prove the claim, we only need to show that the algorithm does not output any element before encountering the secretary. Let i^* be the position in π where the secretary occurs and consider any position $i \in (\tau_1, i^*)$. It must be the case that $\pi(j) \succ_{\mathcal{P}_L} \pi(i)$ for some $j < i$, since otherwise the algorithm would have output $\pi(i)$ when run on $\mathcal{P}_L$. Therefore, either $\pi(j) \succ_{\mathcal{P}} \pi(i)$ or $\pi(j) \|_{\mathcal{P}} \pi(i)$. In both cases, the element $\pi(i)$ is not output when the algorithm runs on $\mathcal{P}$: indeed, in the former case, $\pi(i) \notin \mathcal{F}_{\mathcal{P}}(S_i)$ and in the latter $|\mathcal{F}_{\mathcal{P}}(S_i)| \ge 2 > k$, where $S_i = \bigcup_{j \le i} \{\pi(j)\}$. $\square$

3.2 Analysis for general k

In this section we show that the algorithm succeeds with increasing probability as k increases.

THEOREM 3. *For any poset $\mathcal{P}$ with k maximal elements, Algorithm 1 succeeds with probability at least*

$$\frac{\binom{\tau_k}{k}}{\binom{n}{k}} \cdot \left(\left(1 + \log \frac{n-k}{\tau_k} \right)^k - 1 \right).$$

Before proceeding further, let us briefly interpret the above bound. Let $k = \epsilon n$, and consider a slightly different threshold $\tau'_k = (1-\epsilon)\tau_k = (1-\epsilon)nk^{-\frac{1}{k-1}}$. Note that holding k fixed and letting $n \to \infty$, we have $\epsilon \to 0$.

Then, we can bound $\binom{\tau'_k}{k} / \binom{n}{k}$ as:

$$\frac{\prod_{i=0}^k (1-\epsilon)nk^{-\frac{1}{k-1}}}{\prod_{i=0}^k (n-i)} > \left(\frac{(1-\epsilon)k^{-\frac{1}{k-1}} - \epsilon}{(1-\epsilon)} \right)^k \ge k^{-\frac{k}{k-1}} - O(\epsilon).$$

And,

$$\left(1 + \log \frac{n-k}{\tau'_k}\right)^k = \left(1 + \log \frac{(1-\epsilon)nk^{\frac{1}{k-1}}}{(1-\epsilon)n}\right)^k = \left(1 + \frac{1}{k-1}\log k\right)^k \ge k(1 - o_k(1)).$$

Combining the two, we obtain that the probability of winning is at least

$$\left(k^{-\frac{k}{k-1}} - O(\epsilon)\right) k(1 - o_k(1)) = (1 - o_k(1))k^{-\frac{1}{k-1}} - O(\epsilon).$$

As we will show in Section 4, the term $k^{-\frac{1}{k-1}}$ is tight and thus the maximum difference between the lower bound and the upper bound approaches 0 for large k (Figure 1).

3.2.1 Proof of Theorem 3

To prove the theorem, we will describe the set of permutations on which the algorithm is guaranteed to succeed. In particular, we focus on the probability that the algorithm does not return an element before reaching the last secretary in the permutation. Observe that the algorithm will never pass on the last secretary: it will surely be in $\mathcal{F}_{\mathcal{P}}(S)$, and at that point $\mathcal{F}_{\mathcal{P}}(S) = k$. Obviously, the algorithm will fail if all of the secretaries come before the threshold. A harder to analyze failure mode is that of returning a *faux*-secretary: an element that looks like a maximal element before reaching the actual secretary that dominates it. A way to avoid it is to insist that either $\mathcal{F}_{\mathcal{P}}$ is of size at least $k+1$ before the last secretary is reached or that the maximal element comes before any of the potential faux-secretaries (the latter is exactly the analysis in the $k = 1$ case).

We begin by describing the permutations on which Algorithm 1 will succeed. We first give few definitions that we use in the proof.

Let $\tau = \tau_k$. Fix any $k+1$ special positions in the permutation $1 \le \ell_0 \le \cdots \le \ell_k \le n$, such that for some $0 \le i^* < k$ we have $\ell_{i^*} = \tau$, and the other positions are all distinct from each other. We define the set $P = P_{\ell_0,\ldots,\ell_k}$ of all permutations such that the positions ℓ_i with $i \ne i^*$ are occupied by the secretaries in any order.

For a suffix t_i of elements from position ℓ_i to n, we define $P(t_i)$ be the set of all permutations in P that have t_i as a suffix. For a set of suffixes T, we let $P(T) = \bigcup_{t \in T} P(t)$.

We will now inductively define a set T_{i^*} of suffixes such that Algorithm 1 returns a maximal element in all permutations in $P(T_{i^*})$. To begin, let T_k be the set of all suffixes from ℓ_k to n; thus we have that $P(T_k)$ contains all the permutations in P. Inductively, define T_i, for $i = k-1, \dots, i^*$, in the following way. Let the $\mathcal{F}_{\mathcal{P}}(U \setminus t_{i+1})$ be Pareto frontier of the elements that are *not* in t_{i+1}, and define $\mathcal{G}(t_{i+1})$ to be the set of non-secretary elements of $\mathcal{F}_{\mathcal{P}}(U \setminus t_{i+1})$.

Now let $B(t_{i+1})$ be any subset of $\mathcal{G}(t_{i+1})$ of $\min\{k - i, |\mathcal{G}(t_{i+1})|\}$ elements; we call the set $B(t_{i+1})$ a *blocking set.* Note that in a permutation π where all of the elements in $B(t_{i+1})$ come before ℓ_i, Algorithm 1 cannot terminate with any element between ℓ_i and ℓ_{i+1}. In this case, we say that $B(t_{i+1})$ is a *good* blocking set in π. In order to find a lower bound on the number of winning permutation for a fixed position of i^*, we can bound the number of permutations where $B(t_{j+1})$ is a good blocking set for every $i^* \le j < k$. To this end, let $A(t_{i+1})$ be the set of all suffixes from ℓ_i to n that agree with t_{i+1} and that do not contain elements from $B(t_{i+1})$. Let $T_i = \bigcup_{t_{i+1} \in T_{i+1}} A(t_{i+1})$.

Lemma 4. *Algorithm 1 returns a maximal element on all permutations in $P(T_{i^*})$.*

Proof. Suppose not and consider any permutation in $P(T_{i^*})$ where the algorithm fails. Suppose the returned element is in the interval between ℓ_i and ℓ_{i+1} for some $i^* \le i \le k-1$. Then, this permutation has to be in $P(T_i)$ (since $P(T_i) \subseteq P(T_{i+1})$). But by definition of T_i, if $\mathcal{G}$ is the Pareto frontier of the elements before ℓ_{i+1}, either all of $\mathcal{G}$ or a subset of at least $k-i$ elements of $\mathcal{G}$ comes before ℓ_i. Either way, in the positions $t_i, \dots, t_{i+1}$, the Pareto frontier is composed of at least i secretaries and the good blocking set B_{i+1}. Thus the **if** statement in step 5 of Algorithm 1 avoids that an element is returned in the interval between ℓ_i and ℓ_{i+1}. So we have a contradiction. $\square$

Suppose there were j secretaries that came after the threshold. In this case, we can look at the fraction of the permutations whose suffix agrees with T_{k-j}. Let

$$\gamma(j) = \frac{|P(T_{k-j})|}{|P|}.$$

Note that γ is implicitly a function of $\ell_k, \dots, \ell_{i^*}$. We begin by bounding $\gamma(j)$ from below. Let

$$\gamma'(j) = \prod_{i=k-j}^{k-1} \left(\prod_{w=0}^{k-i-1} \left(\frac{\ell_i - i - w}{\ell_{i+1} - (i+1) - w} \right) \right).$$

Lemma 5. $\gamma(j) \ge \gamma'(j)$.

Proof. By definition, $P(T_i) \subseteq P(T_{i+1})$ and hence

$$\begin{aligned}
|P(T_i)| &= |P(T_{i+1})| \cdot \big(\text{fraction of permutation} \\
&\qquad \text{in } P(T_{i+1}) \text{ with good } B_{t_{i+1}}\big) \\
&\ge |P(T_{i+1})| \cdot \big(\text{fraction of permutation} \\
&\qquad \text{in } P(T_{i+1}) \text{ with } k \text{ elements of} \\
&\qquad \mathcal{G}(t_{i+1}) \text{ before } k-i-1\big) \\
&\ge |P(T_{i+1})| \prod_{w=0}^{k-i-1} \left(\frac{\ell_i - i - w}{\ell_{i+1} - (i+1) - w} \right).
\end{aligned}$$

Therefore, we can conclude that

$$|P(T_{i^*})| \ge |P| \prod_{i=i^*}^{k-1} \left(\prod_{w=0}^{k-i-1} \left(\frac{\ell_i - i - w}{\ell_{i+1} - (i+1) - w} \right) \right).$$

And,

$$\begin{aligned}
\gamma(k - i^*) &= \frac{|P(T_{i^*})|}{|P|} \\
&\ge \prod_{i=i^*}^{k-1} \left(\prod_{w=0}^{k-i-1} \left(\frac{\ell_i - i - w}{\ell_{i+1} - (i+1) - w} \right) \right) \\
&= \gamma'(k - i^*). \quad \square
\end{aligned}$$

We first show that $\gamma'(r)$ can be rewritten in a more convenient way; the proof is in Appendix A.

Lemma 6.

$$\gamma'(r) = \frac{(\ell_{k-r} - (k-r))!}{(\ell_{k-r} - k)!} \prod_{s=0}^{r-1} \frac{1}{\ell_{k-s} - k}.$$

Next, we obtain an analytical bound that will be useful later; the proof is in Appendix B.

Lemma 7.

$$\sum_{\substack{\ell_k, \dots, \ell_{k-j+1}: \\ n \ge \ell_k > \dots > \ell_{k-j} = \tau}} \prod_{s=0}^{j-1} \frac{s+1}{\ell_{k-s} - k} \ge \log^j \frac{n-k}{\ell_{k-j} - (k-j)}.$$

Now we are ready to put all of the pieces together. To count the total number of permutations on which the algorithm succeeds, we begin by conditioning on the number of secretaries that come after the specified threshold, τ. Let E_j be the event such that there are exactly $j \ge 1$ fixed secretaries after the threshold $\ell_{k-j} = \tau$ and let WIN be the event of the algorithm returning a maximal element.

Lemma 8.

$$\Pr[\text{WIN}|E_j] \ge \frac{(\tau - (k-j))!}{(\tau - k)!} \frac{(n - \tau - j)!}{(n - \tau)!} \log^j \left(\frac{n-k}{\tau - (k-j)} \right).$$

Proof. We enumerate over all permutations that have j maximal elements after the threshold. Since $\gamma(j)$ depends only on the position and not on the order of these elements, we have:

$$\Pr[\text{WIN}|E_j] = j! \cdot \sum_{\substack{\ell_k, \dots, \ell_{k-j+1}: \\ n \ge \ell_k > \dots > \ell_{k-j} = \tau}} \left(\frac{1}{n-\tau} \cdots \frac{1}{n - \tau - (j-1)} \right) \gamma(j).$$

Since $\ell_{k-j} = \tau$, applying Lemma 5, Lemma 6, and Lemma 7 completes the proof.

$$\begin{aligned}
&\Pr[\text{WIN}|E_j] \\
&\ge \frac{(\tau - (k-j))!}{(\tau - k)!} \frac{(n - \tau - j)!}{(n - \tau)!} j! \\
&\qquad \cdot \sum_{\substack{\ell_k, \dots, \ell_{k-j+1}: \\ n \ge \ell_k > \dots > \ell_{k-j} = \tau}} \prod_{s=0}^{j-1} \frac{1}{\ell_{k-s} - k} \\
&= \frac{(\tau - (k-j))!}{(\tau - k)!} \frac{(n - \tau - j)!}{(n - \tau)!} \\
&\qquad \cdot \sum_{\substack{\ell_k, \dots, \ell_{k-j+1}: \\ n \ge \ell_k > \dots > \ell_{k-j} = \tau}} \prod_{s=0}^{j-1} \frac{s+1}{\ell_{k-s} - k} \\
&\ge \frac{(\tau - (k-j))!}{(\tau - k)!} \frac{(n - \tau - j)!}{(n - \tau)!} \\
&\qquad \cdot \log^j \left(\frac{n-k}{\tau - (k-j)} \right). \quad \square
\end{aligned}$$

Proof of Theorem 3. Finally, we can remove the conditioning in Lemma 8 to prove an overall bound on the success probability of the algorithm.

$$\begin{aligned}\Pr[E_j] &= \binom{k}{j}\left(\frac{\tau}{n}\cdots\frac{\tau-(k-j-1)}{n-(k-j-1)}\right)\\ &\quad\cdot\left(\frac{n-\tau}{n-(k-j)}\cdots\frac{n-\tau-(j-1)}{n-(k-1)}\right)\\ &= \binom{k}{j}\frac{\tau!}{(\tau-(k-j))!}\cdot\frac{(n-\tau)!}{(n-\tau-j)!}\cdot\frac{(n-k)!}{n!}.\end{aligned}$$

Now, using Lemma 8, we have that

$$\begin{aligned}\Pr[\text{WIN}] &= \sum_{j=1}^{k}\Pr[\text{WIN}|E_j]\Pr[E_j]\\ &= \frac{\tau!}{(\tau-k)!}\cdot\frac{(n-k)!}{n!}\cdot\sum_{j=1}^{k}\binom{k}{j}\log^j\frac{n-k}{\tau-(k-j)}\\ &\geq \frac{\tau!}{(\tau-k)!}\cdot\frac{(n-k)!}{n!}\cdot\sum_{j=1}^{k}\binom{k}{j}\log^j\frac{n-k}{\tau}\\ &= \frac{\tau!}{(\tau-k)!}\cdot\frac{(n-k)!}{n!}\cdot\left(\left(1+\log\frac{n-k}{\tau}\right)^k-1\right).\quad\square\end{aligned}$$

4. UPPER BOUNDS ON SUCCESS

In this section we prove an upper bound on the success probability of any algorithm for the poset secretary problem. For $k = 1$, it is well-known that no algorithm can succeed with probability more than $1/e$. Here we explore how the bound grows with k. Our main result is the following:

Theorem 9. *Let $2 \leq k = o(\sqrt{n})$. For any poset $\mathcal{P}$ with k maximal elements, every algorithm has success probability at most $k^{-\frac{1}{k-1}} + o(1)$.*

To prove this result we will analyze the performance of any algorithm on a specific poset $\mathcal{P}_k$. Let $\mathcal{L}$ be a total order on n/k elements; we will call such a poset a *line*. We define $\mathcal{P}_k$ to be the poset consisting of k disjoint lines: $\mathcal{P}_k = \{\mathcal{L}_1, \ldots, \mathcal{L}_k\}$.

Our strategy is to write down a linear program whose value is an upper bound on the success probability of any algorithm. We will then analyze the dual formulation and derive a feasible solution for it, which will serve as the bound in Theorem 9.

We begin by restricting the class of algorithms and the class of permutations we consider. A *τ-threshold* algorithm is one that never returns any of the first τ elements. Recall that $S_i = \cup_{j<i}\{\pi(j)\}$ denotes the set of elements preceding i in the permutation and $\mathcal{F}(S_i)$ denotes the set of maximal elements of S_i. We insist that the algorithms we consider are *sane*, i.e., they never knowingly return a dominated element; formally, if the element at position i is returned by the algorithm, then $\pi(i) \in \mathcal{F}(S_{i+1})$.

We also restrict the permutations under consideration. A permutation π is called *τ-covering* if the following two conditions hold:

1. $\mathcal{F}(S_\tau) \cap \mathcal{F}_{\mathcal{P}} = \emptyset$, i.e., π has no maximal elements in the first τ positions; and
2. for any $1 \leq j \leq k$, $\mathcal{F}(S_\tau) \cap \mathcal{L}_j \neq \emptyset$, i.e., at least one descendant of each maximal element occurs among the first τ elements.

These restrictions on the algorithm and the permutations do not change the success probability substantially.

Lemma 10. *Consider any algorithm $\mathcal{A}$ that succeeds with probability ρ. Then $\mathcal{A}$ succeeds with probability at least $\rho - o(1)$ on all $(2k\log n)$-covering permutations. Furthermore, when run on $(2k\log n)$-covering permutations, $\mathcal{A}$ is a sane and $(2k\log n)$-threshold algorithm without loss of generality.*

Proof. To prove the first claim, observe that $(2k\log n)$-covering permutations constitute an

$$O\left(\left(1-\frac{1}{k}\right)^{2k\log n} + \left(1-\frac{2k\log n}{n}\right)^k\right) = o(1)$$

fraction of all of the permutations, when $k = o(\sqrt{n})$. Moreover, on these permutations, any algorithm returning one of the first $2k\log n$ elements is guaranteed to fail, therefore we can assume that the algorithm is a $(2k\log n)$-threshold algorithm without loss of generality. $\square$

For the remainder of the proof we therefore assume that the algorithms under consideration are sane and $(2k\log n)$-threshold. We proceed by writing down a linear program that encodes the success probability of any algorithm on $\mathcal{P}_k$.

Lemma 11. *Consider any optimal solution of the linear program in Fig. 2 and let v be its value. Then, any sane $(2k\log n)$-threshold algorithm $\mathcal{A}$, has success probability at most v on the poset $\mathcal{P}_k$.*

Proof. Let $p_i = \Pr[\mathcal{A} \text{ returns } \pi(i)]$ denote the probability that $\mathcal{A}$ returns the ith element of the permutation[1]. Similarly, let $q_i = \Pr[\mathcal{A} \text{ returns } \pi(i) | \pi(i) \in \mathcal{F}(S_{i+1})]$. Note that since $\mathcal{A}$ is sane, $\mathcal{A}$ never returns $\pi(i)$ if $\pi(i) \notin \mathcal{F}(S_{i+1})$. Therefore we can write $p_i = q_i \Pr[\pi(i) \in \mathcal{F}(S_{i+1})]$. Moreover, $\mathcal{A}$ returns an element $\pi(i)$ only if it discards all of the elements in positions $j < i$. Thus we can write:

$$q_i \leq 1 - \sum_{j<i} p_j = 1 - \sum_{j<i} f_j q_i, \tag{1}$$

where $f_j = \Pr[\pi(j) \in \mathcal{F}(S_{j+1})]$.

We can express the probability that $\mathcal{A}$ returns a maximal element as

$$\begin{aligned}&\Pr[\mathcal{A} \text{ wins}]\\ &= \sum_{j=1}^{k}\sum_{i=1}^{n}\Pr[\mathcal{A} \text{ returns } \pi(i)|\pi(i) \in \mathcal{F}(\mathcal{L}_j)]\\ &\quad\cdot\Pr[\pi(i) \in \mathcal{F}(\mathcal{L}_j)]\\ &= \frac{k}{n}\sum_{i=1}^{n}\Pr[\mathcal{A} \text{ returns } \pi(i)|\pi(i) \in \mathcal{F}(S_{i+1})]\\ &= \frac{k}{n}\sum_{i=1}^{n} q_i,\end{aligned}$$

where the second step follows because the algorithm cannot determine whether a maximal element of the poset induced from the first i elements in π is a maximal element

[1]The probability is over both the permutations and the coins of the algorithm.

$$\max_{q_1,\dots,q_n} \frac{k}{n}\sum_{i=1}^{n} q_i$$
$$q_i + \sum_{j=1}^{i-1}\frac{k}{j}q_j \le 1, \qquad 1 \le i \le n$$
$$q_i \ge 0, \qquad 1 \le i \le n$$

$$\min_{x_1,\dots,x_n} \sum_{i=1}^{n} x_i$$
$$x_i + \frac{k}{i}\sum_{j=i+1}^{n} x_j \ge \frac{k}{n}, \qquad 1 \le i \le n$$
$$x_i \ge 0, \qquad 1 \le i \le n$$

Figure 2: Linear program (left) and its dual (right).

of the whole poset or not, and hence the contributions are equal. More formally, we observe that, for every $1 \le j \le k$, $\Pr[\mathcal{A} \text{ returns } \pi(i) | \pi(i) \in \mathcal{F}(\mathcal{P}_k) \cap \mathcal{L}_j] = \Pr[\mathcal{A} \text{ returns } \pi(i) | \pi(i) \in \mathcal{F}(S_{i+1}) \cap \mathcal{L}_j]$. This follows because for any two permutations π, π' identical up to $i-1$, and with $\pi(i) \in \mathcal{F}_\pi(\mathcal{P}_k) \cap \mathcal{L}_j$ and $\pi'(i) \in \mathcal{F}_{\pi'}(S_{i+1}) \cap \mathcal{L}_j$, we have that the poset induced by the first i elements is exactly the same. Hence, the algorithm's behavior is unchanged (here the subscript on $\mathcal{F}$ denotes the permutation of the elements under consideration).

Finally we show that we correctly captured the constraints on q. Assume that $i > 2k \log n$, and denote by S_τ the set of elements appearing before the threshold. Since for all j, $S_\tau \cap \mathcal{L}_j \neq \emptyset$, the size of the Pareto set at i is exactly k. Therefore $f_i = {}^k\!/_i$. For $i > 2k \log n$, inequality (1) implies that $q_i + \sum_{j<i} \frac{k}{j} q_j \le 1$.The same inequality trivially holds when $i \le 2k \log n$ since $q_i = 0$ for these elements. $\square$

Next, we focus on the feasible solution to the dual program.

Lemma 12. *There exists a feasible solution to the dual program in Figure 2 that has value $k^{-\frac{1}{k-1}} + o(1)$.*

Proof. We define the following feasible solution to the dual program in Figure 2: inductively, $x_n = \frac{k}{n}$ and $x_i = \max\{0, \frac{k}{n} - \frac{k}{i}\sum_{j=i+1}^{n} x_j\}$. Note that the value of the objective function for this solution is $\sum_{i=1}^{n} x_i = \sum_{i=T+1}^{n} x_i$, where T is the maximum index such that $x_T = 0$. Consider the sequence a_i inductively defined by $a_n = \frac{k}{n}$ and $a_i = a_{i+1} + \left(\frac{k}{n} - \frac{k}{i}a_{i+1}\right)$. We observe that for any $j \ge T+1$, it holds that $a_j = \sum_{i=j}^{n} x_i$ and that $\frac{k}{n} - \frac{k}{j}a_{T+1} \le 0$. Specifically, either $a_T < a_{T+1}$ and $a_{T+1} > \cdots > a_n$, or $a_{T-1} < a_T = a_{T+1}$ and $a_{T+1} > \cdots > a_n$.

Note that the problem of computing $\sum_{i=1}^{n} x_i$ now reduces to the problem of finding the last local maximum of the sequence $\{a_i\}_{i=1}^{n}$.

We proceed as follows. First, we introduce the function $s(z)$ defined over the real domain $[1, n]$ by

$$s(z) = \frac{k}{n}\sum_{\ell=0}^{n-z}\prod_{t=1}^{\ell}\left(1 - \frac{k}{z+t-1}\right).$$

Note that $s(i) = a_i$ for every $i \in \{1, \dots, n\}$. We study $s(z)$ in $[g(n), n]$ with $g(n) = \omega(1)$ and show that it has only one stationary point (a maximum) in this interval at $z^* = (1 \pm o(1))n/k^{1/(k-1)}$. Finally, since $s(z)$ is continuous, we can conclude that $T + 1 = \max\{\lfloor z^* \rfloor, \lceil z^* \rceil\}$.

We have

$$\begin{aligned}
s(z) &= \frac{k}{n}\sum_{\ell=0}^{n-z}\prod_{t=1}^{\ell}\left(1 - \frac{k}{z+t-1}\right)\\
&= \frac{k}{n}\sum_{\ell=0}^{n-z}\exp\left[\sum_{t=1}^{\ell}\log\left(1 - \frac{k}{z+t-1}\right)\right]\\
&\ge \frac{k}{n}\sum_{\ell=0}^{n-z}\exp\left[\int_{t=1}^{\ell}\log\left(1 - \frac{k}{z+t-1}\right)dt\right]\\
&= \frac{k}{n}\sum_{\ell=0}^{n-z}\exp\left[(z+t-1)\log\left(1 - \frac{k}{z+t-1}\right)\right.\\
&\qquad \left. -k\log(z-1-k+t)\Big|_{t=1}^{\ell}\right]\\
&= \frac{k}{n}\sum_{\ell=0}^{n-z}\frac{\left(1 - \frac{k}{z+\ell-1}\right)^{z+\ell-1}(z-k)^k}{\left(1 - \frac{k}{z}\right)^z (z+\ell-1-k)^k}.
\end{aligned}$$

One can similarly show that

$$s(z) \le \frac{k}{n}\sum_{\ell=0}^{n-z}\frac{\left(1 - \frac{k}{z+\ell}\right)^{z+\ell}(z-k)^k}{\left(1 - \frac{k}{z}\right)^z (z+\ell-k)^k}.$$

For $z \ge k \log n$, we have that $\left(1 - \frac{k}{z+l-1}\right)^{z+l-1} = (1 - o(1))e^{-k}$ and $\left(1 - \frac{k}{z}\right)^z = (1 - o(1))e^{-k}$, where the $o(1)$ term hides factors going to 0 as $n \to \infty$. So we can write $s(z)$ as

$$\begin{aligned}
s(z) &\ge (1 - o(1))\frac{k}{n}(z-k)^k\sum_{\ell=0}^{n-z}\frac{1}{(z+\ell-1-k)^k}\\
&\ge (1 - o(1))\frac{k}{n}(z-k)^k\int_{\ell=0}^{n-z}\frac{1}{(z+\ell-1-k)^k}d\ell\\
&= (1 - o(1))\frac{k}{n}\frac{1}{k-1}(z-k)^k\\
&\qquad \left(\frac{1}{(z-k-1)^{k-1}} - \frac{1}{(n-k-1)^{k-1}}\right)\\
&\ge (1 - o(1))\frac{k}{n}\frac{1}{k-1}\left((z-k) - \frac{(z-k)^k}{(n-k-1)^{k-1}}\right).
\end{aligned}$$

One can similarly show that

$$s(z) \le (1 + o(1))\frac{k}{n}\frac{1}{k-1}\left((z-k) - \frac{(z-k)^k}{(n-k-1)^{k-1}}\right).$$

Taking the derivative and setting it to zero gives a maximum at z^* at $\frac{n}{k^{1/(k-1)}}(1 \pm o(1))$. Define $i^* = \max\{\lfloor z^* \rfloor, \lceil z^* \rceil\}$. By definition of a_i and the maximality of i^*, it must be that

$\frac{k}{n} - \frac{k}{i^*} a_{i^*+1} \geq 0$, which implies

$$a_{i^*+1} \leq \frac{i^*}{n} \leq k^{-1/(k-1)}(1 + o(1)).$$

Finally,

$$\sum_{i=1}^{n} x_i = a_{i^*} \leq a_{i^*+1} + \frac{k}{n} \leq k^{-1/(k-1)} + o(1).$$

This gives a feasible solution to the dual and a bound on its value. □

5. TIGHTNESS OF ALGORITHM 1

In Section 4 we showed that no algorithm can succeed with probability more than $k^{-\frac{1}{k-1}} - o(1)$ on the poset $\mathcal{P}_k$ consisting of k disjoint total orders. In this section we use a different analysis to show that Algorithm 1 achieves this bound on a large family of posets including $\mathcal{P}_k$.

The main idea is to define an event of the algorithm passing on a maximal element after the threshold τ_k and then returning a non-maximal element; call this event PASS. For any $\tau_k < i \leq n$, we define A_i as the event that the first secretary after the threshold occurs at position i and D_i as the event that the algorithm discards the element $\pi(i)$; let $D_{i_1:i_2} = D_{i_1} \wedge \cdots \wedge D_{i_2}$. Now, if WIN denotes the event that the algorithm returns a secretary, then PASS $= \cup_{i>\tau_k} \text{PASS}_i$, where $\text{PASS}_i = D_{\tau_k+1:i} \wedge A_i \wedge \overline{\text{WIN}}$.

We begin by showing that the disjoint events WIN and PASS together account for the vast majority of the outcomes of the algorithms. We then show how to upper bound the probability of PASS on particular posets, leading to a bound on the success probability of the algorithm.

LEMMA 13. $\Pr[\text{WIN}] + \Pr[\text{PASS}] \geq \frac{k}{n} \sum_{i=\tau_k+1}^{n} \frac{\binom{\tau_k}{k}}{\binom{i-1}{k}}$.

PROOF. Suppose the first secretary after the threshold is at i. Then the algorithm can win by returning $\pi(i)$ or discarding $\pi(i)$ and then winning later on. In the first case, it must be that $\mathcal{F}(S_{i+1}) \leq k$ while in the other, $\mathcal{F}(S_{i+1}) \geq k+1$. Let F_i denote the event that $|\mathcal{F}(S_{i+1})| \geq k+1$. Then, we have

$$\begin{aligned}\Pr[\text{WIN}] = \sum_{i>\tau_k} \big(&\Pr[D_{\tau_k+1:i-1} \wedge A_i \wedge \overline{F_i}] \\ &+ \Pr[D_{\tau_k:i-1} \wedge A_i \wedge F_i \wedge \text{WIN}]\big).\end{aligned}$$

The second term can be rewritten as

$$\begin{aligned}&\Pr[D_{r_k:i-1} \wedge A_i \wedge F_i \wedge \text{WIN}] \\ &\quad = (1 - \Pr[\overline{\text{WIN}} | D_{r_k:i-1} \wedge A_i \wedge F_i]) \\ &\qquad \cdot \Pr[D_{r_k:i-1} \wedge A_i \wedge F_i] \\ &\quad = \Pr[D_{r_k:i-1} \wedge A_i \wedge F_i] - \Pr[\text{PASS}_i].\end{aligned}$$

Since the events PASS_i are disjoint, we have that $\Pr[\text{WIN}] + \Pr[\text{PASS}] = \sum_{i=\tau_k+1}^{n} \Pr[D_{\tau_k:i-1} \wedge A_i]$. We proceed to bound $\Pr[D_{\tau_k:i-1} \wedge A_i]$. Note that the probability of $\pi(i)$ being a secretary is k/n. Consider any subset B of $\mathcal{F}(S_i)$ that contains all secretaries in $\mathcal{F}(S_i)$ and is of size $\min\{|\mathcal{F}(S_i)|, k\}$. B is a blocking subset: if all elements in B appear before the threshold, no element between $\tau_k + 1$ and $i - 1$ (inclusive) will be accepted. Moreover, this implies that there are no maximal elements between $\tau_k + 1$ and $i - 1$. The lemma follows observing that the probability of this event is at least $\frac{\binom{\tau_k}{\ell}}{\binom{i-1}{k}}$. □

We can now prove a concrete bound on the winning probability that closely resembles the bound in Section 4.

THEOREM 14. *For any k,*

$$\Pr[\text{WIN}] \geq k^{-\frac{1}{k-1}} - \Pr[\text{PASS}] - O(k/n).$$

PROOF. By Lemma 13,

$$\begin{aligned}&\Pr[\text{WIN}] + \Pr[\text{PASS}] \\ &\geq \frac{k}{n} \sum_{i=r_k+1}^{n} \frac{\tau_k \cdots (\tau_k - k + 1))}{(i-1)\cdots(i-k)} \\ &\geq \frac{k}{n}(\tau_k - k + 1)^k \cdot \sum_{i=\tau_k+1}^{n} \frac{1}{(i-k)^k} \\ &\geq \frac{k}{n}(\tau_k - k + 1)^k \cdot \int_{i=\tau_k+1}^{n+1} \frac{1}{(i-k)^k} \\ &= \frac{k}{n}(\tau_k - k + 1)^k \frac{1}{k-1} \\ &\qquad \left(\frac{1}{(\tau_k - k + 1)^{k-1}} - \frac{1}{(n-k+1)^{k-1}}\right) \\ &= \frac{\tau_k - k + 1}{n} \frac{k}{k-1} \left(1 - \left(\frac{\tau_k - k + 1}{n - k + 1}\right)^{k-1}\right) \\ &\geq \frac{\tau_k - k + 1}{n} \frac{k}{k-1} \left(1 - \left(\frac{\tau_k}{n}\right)^{k-1}\right).\end{aligned}$$

Now, by definition, we have $\tau_k = n/k^{\frac{1}{k-1}}$. Therefore,

$$\Pr[\text{WIN}] + \Pr[\text{PASS}] \geq \frac{\tau_k - k + 1}{n} = k^{-\frac{1}{k-1}} - \frac{k+1}{n},$$

which concludes the proof. □

Note that the only way for the algorithm to discard a secretary $\pi(i)$ with $i > \tau_k$ is if $|\mathcal{F}(S_{i+1})| > k$. This event has zero probability in the poset $\mathcal{P}_k$ consisting of k disjoint total orders.

COROLLARY 15. *Algorithm 1 succeeds with probability $k^{-\frac{1}{k-1}} - O(k/n)$ on any poset of width at most k. In particular, this holds true for the poset $\mathcal{P}_k$.*

6. REFERENCES

[1] M. Ajtai, N. Megiddo, and O. Waarts. Improved algorithms and analysis for secretary problems and generalizations. *SIAM J. Discrete Math.*, 14(1):1–27, 2001.

[2] M. Babaioff, M. Dinitz, A. Gupta, N. Immorlica, and K. Talwar. Secretary problems: Weights and discounts. In *Proc. 20th SODA*, pages 1245–1254, 2009.

[3] M. Babaioff, N. Immorlica, and R. Kleinberg. Matroids, secretary problems, and online mechanisms. In *Proc. 18th SODA*, pages 434–443, 2007.

[4] Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. Online auctions and generalized secretary problems. *SIGecom Exchanges*, 7(2), 2008.

[5] F. T. Bruss. Sum the odds to one and stop. *Annals of Probability*, 28:1384–1391, 2000.

[6] N. Buchbinder, K. Jain, and M. Singh. Secretary problems via linear programming. In *Proc. 14th IPCO*, pages 163–176, 2010.

[7] E. B. Dynkin. The optimum choice of the instant for stopping a Markov process. *Sov. Math. Dokl.*, 4:627–629, 1963.

[8] T. Ferguson. Who solved the secretary problem. *Statist. Sc.*, 4:282–296, 1989.

[9] R. Freij and J. Wästlund. Partially ordered secretaries. *Electronic Communication in Probability*, 15:504–507, 2010.

[10] N. Georgiou, M. Kuchta, M. Morayne, and J. Niemiec. On a universal best choice algorithm for partially ordered sets. *Random Struct. Algorithms*, 32(3):263–273, 2008.

[11] A. V. Gnedin. Multicriteria extensions of the best choice problem: Sequential selection without linear order. *Contemp. Math*, 125:153–172, 1992.

[12] A. Gupta, A. Roth, G. Schoenebeck, and K. Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *Proc. 6th WINE*, pages 246–257, 2010.

[13] M. T. Hajiaghayi, R. Kleinberg, and D. C. Parkes. Adaptive limited-supply online auctions. In *Proc. 5th EC*, pages 71–80, 2004.

[14] R. Kleinberg. A multiple-choice secretary problem with applications to online auctions. In *Proc. 16th SODA*, pages 630–631, 2005.

[15] J. Kozik. Dynamic threshold strategy for universal best choice problem. In *Proc. 21st International Meeting on Probabilistic, Combinatorial, and Asymptotic Methods in the Analysis of Algorithms*, pages 439–452, 2010.

[16] G. Kubicki, J. Lehel, and M. Morayne. A ratio inequality for binary trees and the best secretary. *Combinatorics, Probability, and Computing*, 11:146–161, 2002.

[17] M. Zadimoghaddam M. H. Bateni, M. T. Hajiaghayi. Submodular secretary problem and extensions. In *Proc. 6th WINE*, pages 39–52, 2010.

[18] M. Morayne. Partial-order analogue of the secretary problem; the binary tree case. *Discrete Math*, 184:165–181, 1998.

[19] J. Preater. The best-choice problem for partially ordered objects. *Oper. Res. Lett*, 25:187–190, 1999.

[20] J. A. Soto. Matroid secretary problem in the random assignment model. In *Proc. 22nd SODA*, pages 1275–1284, 2011.

APPENDIX

A. PROOF OF LEMMA 6

Proof. We prove the statement by induction. By inspection, the equality holds for $r = 1$. Now suppose it holds up to a certain $1 \le r < k$. Then, we have

$$\begin{aligned}
\gamma'(r+1) &= \prod_{i=k-r}^{k-1}\left(\prod_{w=0}^{k-i-1}\left(\frac{\ell_i - i - w}{\ell_{i+1}-(i+1)-w}\right)\right)\\
&= \gamma'(r)\cdot\prod_{w=0}^{r}\left(\frac{\ell_{k-(r+1)}-(k-(r+1))-w}{\ell_{k-r}-(k-r)-w}\right)\\
&= \frac{(\ell_{k-r}-(k-r))!}{(\ell_{k-r}-k)!}\left(\prod_{s=0}^{r-1}\frac{1}{\ell_{k-s}-k}\right)\\
&\quad\cdot\frac{(\ell_{k-r}-(k-r)-(r+1))!}{(\ell_{k-r}-(k-r))!}\\
&\quad\cdot\frac{(\ell_{k-(r+1)}-(k-(r+1)))!}{(\ell_{k-r+1}-(k-(r+1))-(r+1))!}\\
&= \frac{(\ell_{k-(r+1)}-(k-(r+1)))!}{(\ell_{k-r+1}-k)!}\cdot\frac{1}{\ell_{k-r}-k}\\
&\quad\cdot\left(\prod_{s=0}^{r-1}\frac{1}{\ell_{k-s}-k}\right),
\end{aligned}$$

which concludes the proof. □

B. PROOF OF LEMMA 7

Proof. For convenience, define ϕ as $\phi(k+1) = 1$ and for $r \le k$,

$$\phi(r) = \sum_{\ell_r:\ell_r=\ell_{r-1}+1}^{n-(k-r)}\frac{k-r+1}{\ell_r-k}\phi(r+1).$$

It is easy to see that

$$\sum_{\substack{\ell_k,\dots,\ell_{k-j+1}:\\ n\ge\ell_k>\dots>\ell_{k-j}=\tau}}\prod_{s=0}^{j-1}\frac{s+1}{\ell_{k-s}-k} = \phi(k-(j-1)).$$

We now provide a lower bound on $\phi(r)$.

$$\phi(r) \ge \log^{k-r+1}\frac{n-k}{\ell_{r-1}-(r-1)}.$$

We proceed by backward induction on r. For $r = k+1$, the claim holds trivially. Suppose the claim holds down to a certain $r+1$. Then,

$$\begin{aligned}
\phi(r) &= \sum_{\ell_r=\ell_{r-1}+1}^{n-(k-r)}\frac{k-r+1}{\ell_r-k}\phi(r+1)\\
&= \sum_{\ell_r=\ell_{r-1}+1}^{n-(k-r)}\frac{k-r+1}{\ell_r-k}\log^{k-r}\frac{n-k}{\ell_r-r}\\
&\ge \sum_{\ell_r=\ell_{r-1}+1}^{n-(k-r)}\frac{k-r+1}{\ell_r-r}\log^{k-r}\frac{n-k}{\ell_r-r}\\
&\ge \int_{\ell_{r-1}+1}^{n-(k-r)}\frac{k-r+1}{x-r}\log^{k-r}\frac{n-k}{x-r}\\
&= -\log^{k-r+1}\frac{n-k}{x-r}\Bigg|_{x=\ell_{r-1}+1}^{n-(k-r)}\\
&= \log^{k-r+1}\frac{n-k}{\ell_{r-1}-(r-1)}. \quad\square
\end{aligned}$$