Secretary Problems: Weights and Discounts

Moshe Babaioff* Michael Dinitz† Anupam Gupta‡ Nicole Immorlica§
Kunal Talwar¶

Abstract

The classical secretary problem [Dyn63, Fre83] studies how to select online an element with maximum value in a randomly ordered sequence. The problem is closely connected with online mechanism design in which agents $\{e\}$ with private values $v(e)$ for a good arrive sequentially in random order and the mechanism designer wishes to allocate the good to an agent with maximum value. The difficulty lies in the fact that an agent's allocation must be decided irrevocably upon arrival. A mechanism for this problem is called α-competitive if it gets, in expectation, at least a $1/\alpha$ fraction of the (expected) optimal offline solution. It is well-known how to design constant-competitive algorithms for the classical secretary problems and several variants. We study the following two extensions:

- In the *discounted secretary problem*, there is a time-dependent "discount" factor $d(t)$, and the benefit derived from assigning the good at time t to agent e is $d(t) \cdot v(e)$. For this problem, we show a lower bound of $\Omega(\frac{\log n}{\log \log n})$, and a nearly-matching $O(\log n)$-competitive algorithm with general (and possibly increasing) $d(t)$. Additionally, we show a constant-competitive algorithm when the expected optimum is known in advance.
- In the *weighted secretary problem*, K goods are available and each good k has a publicly known characteristic or weight $w(k)$ such that the value of agent e for good k is $w(k) \cdot v(e)$. The goal is to assign goods to maximize $\sum_{e,k} w(k) \cdot v(e) \cdot x_{ek}$ where x_{ek} is an indicator variable that agent e received good k. We give constant-competitive algorithms for this problem.

One can of course also consider a setting in which there are both discounts and weights. We show that our algorithms can generally be combined and extended to handle this mixed scenario.

Most of these results can also be extended to the *matroid secretary* case [BIK07] with a constant-factor loss for a large family of matroids, and an $O(\log \text{rank})$ loss for general matroids. These results are based on a reduction from various matroids to partition matroids which present a unified approach to many of the upper bounds of [BIK07].

*moshe@microsoft.com. Microsoft Research, Silicon Valley.
†mdinitz@cs.cmu.edu. Carnegie Mellon University.
‡anupamg@cs.cmu.edu. Carnegie Mellon University.
§immorlic@cwi.nl. Centrum voor Wiskunde en Informatica.
¶kunal@microsoft.com. Microsoft Research, Silicon Valley.

1 Introduction

The classical secretary problem [Dyn63, Fre83] captures the question of finding the element with the maximum value in an *online* fashion, when the elements are presented in a random order. It is well known that waiting until one sees $1/e$ fraction of the elements, and picking the first element attaining a value greater than the maximum value seen in the first $1/e$ fraction of the elements gives an e-competitive algorithm, and this is the best possible. The problem is of much interest due to its close connections with online mechanism design: if we have a single good to sell and agents with varying valuations for that object arriving online (albeit in a random order [1]), then the secretary problem captures the difficulty in picking the person with the largest valuation for that good [HKP04, Kle05]. In this case, the elements of the secretary problem are agents and the element value is the agent value for the good; the goal of the mechanism designer is to *maximize social welfare*, or sell the good to the agent with the highest valuation. The problem can also be thought of as modeling the economic decision facing an agent who wishes to select one of an online sequence of goods – e.g., an agent buying a house or a company hiring an employee. In this case, the elements are the goods and the element value is the value of the agent for the said good.

Given the above interpretations, there are certain natural cases which the secretary problem does not address: for instance, it does not capture the opportunity cost incurred due to rejecting elements. For example, when seeking to purchase a house, we might think of choosing a slightly suboptimal house at the beginning of the experiment (and being able to use it for the entire period) as being more desirable than a long wait to pick the most desirable house. Here, we model such a problem as the *discounted secretary problem*, where we are given "discount" values $d(t)$ for every time: the benefit derived from choosing an element with value $v(e)$ at time t is the product $d(t) \cdot v(e)$. In this example, the discount function $d(t)$ is a monotone decreasing function of t, but in general, the discount function may be more complicated due to other considerations. For example, our financial situation may improve over time

[1] The random order assumption allow us to overcome the impossibility of achieving good competitive ratio in the case of adversarial input without assuming that values come from some distribution.

and waiting longer may get us a better mortgage rate, so our discount function $d(t)$ may increase up to some point in time, and then decrease.[2]

An orthogonal extension of the classical secretary problem is the *weighted secretary problem*. In this case, there are K heterogeneous goods $\{1, 2, \ldots, K\}$ available, with the k^{th} good having a publicly known characteristic or "weight" $w(k)$, representing a common ranking of the goods. Agents' preferences for the goods are related by a multiplicative factor, i.e., each agent e has an intrinsic value $v(e)$ and a corresponding value of $v(e) \cdot w(k)$ for good k. Such so-called "product valuations" are commonly assumed in industries like online banner advertising. Here the weight of the good (banner advertising space, in this case) is its visibility, or the number of people that are likely to see it. The intrinsic value of the advertising company is the value it derives when one person sees its ad. Similarly, product valuations might be observed in hiring scenarios. For example, a company may wish to hire sales managers for several regional markets of varying sizes. The weights of the goods (job positions, in this case) are the market sizes, and the value of a manager is his or her inherent ability to convert peoples' interest into actual sales. As in the classical and the discounted settings the agents (elements) arrive in a random order, and when an agent arrives, we must decide which good to allocate to him or her, if any. Each good can only be assigned to one agent, each agent can only receive one good, and once a good is assigned to an agent, that decision is irrevocable. If we assign good k to agent e we receive benefit $v(e) \cdot w(k)$, the product of the intrinsic value of the agent and the weight of the assigned good. How should we choose K agents and assign goods to them to maximize the total expected benefit?

Our goal is to design algorithms for the above secretary problems that do well with respect to the optimum offline solution for an arbitrary set of values $\{v(e)\}$. We call an algorithm α-competitive ($\alpha \geq 1$) if it gets, in expectation over the random sequence of element arrivals, a $1/\alpha$ fraction of (the expected value of) the offline optimum (see Section 2 for a precise definition). We show that the worst-case competitive ratio for the discounted secretary problem is close to logarithmic:

Theorem 1.1 (Discounted Secretary) *Any algorithm for the discounted secretary problem has a (worst-case) competitive ratio of $\Omega(\frac{\log n}{\log \log n})$. Moreover, there is a nearly-matching $O(\log n)$-competitive algorithm.*

Surprisingly, the discounted secretary problem is interesting even if we know the values of all the items in advance: given discount function $d(t)$, it is not *a-priori* obvious which item to choose in this case. In stark contrast to Theorem 1.1, we show the following.

Theorem 1.2 (Discounted Sec'y: Known-OPT) *The discounted secretary problem has an $O(1)$-competitive algorithm for the case when the values of all elements are known in advance. In fact, the algorithm only needs to know $\mathbf{E}[\mathsf{OPT}]$.*[3]

The assumption that the values of element are known holds, say, when the value is a function of the ordinal preference (e.g., the value of the top candidate among n candidates is n, the second-best candidate has value of $n-1$, and so on), which has been used in [Lin61]. Alternatively, $\mathbf{E}[\mathsf{OPT}]$ may be estimated by market research or from prior runs of the mechanism.

For the weighted secretary problem, we show the following.

Theorem 1.3 (Weighted Secretary) *There is an $O(1)$-competitive algorithm for the weighted secretary problem.*

As the classical secretary problem is a special case of the weighted secretary problem when there is only one non-zero weight, there is clearly a lower-bound of e for the weighted secretary problem.

In the setting with both discounts and weights, we show that a combination of the above algorithms yields a nearly-optimal result (since the $\Omega(\frac{\log n}{\log \log n})$ lower bound still holds).

Theorem 1.4 *There is an $O(\log n)$-competitive algorithm for the secretary problem with both weights on goods and discounts on times.*

Finally, we consider the discounted and weighted versions of the *matroid secretary problem* [BIK07], where the goal is to choose a set of items in order to maximize the total expected value, subject to the constraint that the chosen set is independent in a given matroid (see Section 5 for definitions). We show that the algorithms of Theorems 1.1, 1.2 and 1.3 can be extended to a large family of matroids (including uniform matroids, partition matroids, graphical matroids) with only a constant loss in the competitive ratio, and to all matroids with an $O(\log \text{rank})$ loss in the competitive ratio. These results are based on reductions from various classes of matroids to partition matroids, which also give a unified approach to many of the upper bounds of [BIK07], whilst improving some of them. For example, our techniques imply the following:

[2] In this paper, we use the term "discount" though the function $d(t)$ is not monotone decreasing as a function of t.

[3] $\mathbf{E}[\mathsf{OPT}]$ is the expected benefit the optimal algorithm gets.

Theorem 1.5 *There is a* $3e \approx 8.15$*-competitive algorithm for the matroid secretary problem for graphical matroids. (The previous best known was a* 16*-competitive algorithm [BIK07].)*

While we state our results as algorithms, the setting which motivates us is actually an economic one in which the elements are strategic agents and their values are private information. In this case, it is important to consider the incentives facing agents in our proposed algorithms. Assuming single parameter agents (i.e., agent e has value $v(e)$ if he is picked by the algorithm, and 0 otherwise), it is well known that a mechanism is truthful (in dominant strategies) if it is bid monotonic (a winner would keep winning if she increases her bid). An alternative interpretation is that the mechanism presents the agent with a price that is independent of the agent's bid and the agent decides if she would like to win given the price. This is indeed the case for all our algorithms, thus one can interpret our algorithms as truthful mechanisms in which winners pay the threshold value they needed to bid in order to win. [4] The competitive ratio that an algorithm achieves corresponds to the fraction of the social welfare that the truthful mechanism guarantees.

Related Work. The study of truthful on-line mechanism design in the competitive analysis framework was initiate by Lavi and Nisan [LN00] and many other papers followed this line of research, e.g. [LN05]. The classical secretary problem was introduced by Dynkin in 1963 [Dyn63]. Since then, many variants have been studied (see [Fer89, Fre83] for a survey), including some in the computer science literature highlighting the connections to online mechanism design [HKP04, Kle05] and combinatorial preference structures [BIK07, BIKK07]. The discounted problem was studied previously in multiple contexts for specific "well-behaved" functions like $d(t) = \beta^t$ [RP76] or $d(t) = \sum_{i=t}^{n} \beta^t$ [MMP08] for some fixed $\beta < 1$, whereas here we study it for general functions $d(t)$. For the weighted case, Derman, Lieberman, and Ross [DLR71] studied a version where there are the same number of goods as agents (so $K = n$) and the values of the agents are independently and identically distributed, as opposed to our setting where the values are arbitrary but arrive in random order. Similarly, a recent paper by Gershkov and Moldovanu [GM07] studied the variant where the values of the elements are independently and identically distributed and element arrivals are given by some renewal stochastic process, instead of the classical secretary assumption of discrete time with a uniformly random ordering. They further incorporated the discounted model into the weighted model (so the value of agent e for good k at time t is $v(e) \cdot w(k) \cdot d(t)$, and show how to maximize revenue as well as welfare in their setting).

[4] Note that we do not allow agents to manipulate their arrival time - their order is random and cannot be influenced by the agents. Moreover, each agent is considered only once and cannot return later in time.

2 Model, Preliminaries, and Notation

In the classical secretary problem, there is a universe U of elements with $|U| = n$. Each element $e \in U$ has an intrinsic value $v(e) \in \mathbb{R}_{\geq 0}$. An algorithm A for the secretary problem observes the elements of U in a random order and chooses one element e_A in an *online* fashion. In other words, it must decide at the moment an element arrives whether or not to select it, and all decisions are irrevocable. The goal is to maximize the expected value $\mathbf{E}[v(e_A)]$, the expectation being taken over the random order, as well as over the randomness in the algorithm, if any. In this paper, we extend the classical setting in the following ways:

Weighted Secretary Problems. Here, we want to assign K goods to elements (agents). Each good k has a non-negative associated weight $w(k)$. Without loss of generality we assume that $w(\cdot)$ is non-increasing (i.e., good 1 has the highest weight, and so on). The algorithm assigns goods to agents as the agents arrive, never assigning a good to more than one agent or giving an agent more than one good. In other words, an algorithm A maintains a map $s_A : [K] \to U \cup \{\perp\}$, where we interpret $s_A(k) = \perp$ to mean that the kth good is not assigned to any agent. For ease of notation we extend the agent valuation function by letting $v(\perp) = 0$. This map is almost injective in that no two goods can be assigned to the same agent, so if $s_A(i) = s_A(j) \neq \perp$ then $i = j$.

Agents are observed in a random order, and initially $s(k) = \perp$ for all $k \in [K]$. When an agent e arrives the algorithm finds out $v(e)$, and if there is some unassigned good (i.e. there exists some k such that $s_A(k) = \perp$) the algorithm can choose whether or not to assign good k to agent e. If it decides to give e some currently unassigned good k, then it sets $s_A(k) = e$. The goal of the algorithm is to output a final map s_A that maximizes the objective function $\mathbf{E}[\sum_{k=1}^{K} v(s_A(k))w(k)]$, where the expectation is over the random ordering and the random choices of the algorithm, if any.

Discounted Secretary Problems. Here, the algorithm is given as input a discount function $d : [n] \to \mathbb{R}_{\geq 0}$ which maps "time" to "discounts". Again, the ground set is presented in random order: we formalize this as picking a bijective ordering function $\pi : [n] \to U$ uniformly at random from all bijective functions from time instants $[n]$ to elements U, implying that element $\pi(t) \in U$ appears at time instant t. In this

problem, we want to choose an element e_A in an online fashion to maximize the expected discounted value $\mathbf{E}_\pi[d(\pi^{-1}(e_A)) \cdot v(e_A)]$. We first consider a classical worst-case model where the algorithm does not have any prior knowledge about the input. We also consider the *known-*OPT *model*, where the algorithm knows the expected value of the optimal offline solution ahead of time (for example, it is sufficient to assume the algorithm knows the valuations), it just does not know the random permutation π.

We use the *competitive ratio* of an online algorithm to measure the performance of all the algorithms in this paper: this is just the performance of the algorithm compared to the performance of the best *offline* algorithm. Since we are dealing with maximization problems, an α-competitive algorithm A for the classical secretary problem guarantees that for all valuation functions $v(\cdot)$, $\mathsf{OPT}/\mathbf{E}[v(e_A)]$ is at most α where $\mathsf{OPT} = \max_{e \in U} v(e)$ (note $\alpha \geq 1$). Similarly, an α-competitive algorithm A for the weighted secretary problem guarantees that $\mathsf{OPT}/\mathbf{E}[\sum_{e \in S_A} v(e)w(s_A(e))]$ is at most α where $\mathsf{OPT} = \sum_{i=1}^{K} w(k)v(\sigma(k))$ and σ is the permutation that sorts agents in decreasing order of value. For the discounted secretary problem, we need to be careful in defining the competitive ratio as the optimum solution is now a random variable (the optimum offline solution is a function of the random order of arrivals). We define the competitive ratio of an algorithm A in this case as the smallest value α such that

$$\frac{\mathbf{E}_\pi[\max_e d(\pi^{-1}(e)) \cdot v(e)]}{\mathbf{E}_\pi[d(\pi^{-1}(e_A)) \cdot v(e_A)]} \leq \alpha.$$

It is well-known that the following algorithm is e-competitive for the classical secretary problem: observe a $(1/e)$ fraction of the elements without selecting any. In the remaining $(1-1/e)$ elements, select the first element whose value is greater than all elements preceding it. In the remainder of the paper, we will refer to this algorithm as the classical secretary algorithm, and we will use it (and the sample-then-select intuition behind it) to design algorithms for the weighted and discounted cases.

Finally, all the secretary problems mentioned above can be extended to the *matroid secretary case* where, instead of picking a set S of one or K elements, we are given a matroid $\mathcal{M} = (U, \mathcal{I})$ and want to pick a set $S \in \mathcal{I}$ that maximizes the objective function. Details of this extension, and background on matroids, appear in Section 5.

3 The Weighted Secretary Problem

Recall the weighted secretary problem was to choose, in an online fashion, an assignment of K goods with weights $w(1) \geq w(2) \geq \cdots \geq w(K)$ to agents with unit demand. The goal was to maximize the weighted value $\sum_{k=1}^{K} v(s(k)) \cdot w(k)$, where $s(k)$ is the agent to which good k is assigned and $v(s(k)) = 0$ if k is not assigned to any agent (i.e. $s(k) = \perp$). In this section, we show a constant-competitive algorithm for this problem; in Appendix C, we will extend the algorithm to a variety of matroids as well.

There are two basic cases for the weighted secretary problem. Either there is one agent with very high value, in which case we must assign this agent the highest-weight good, or there are many agents with high value. We deal with each case separately by running the classical secretary algorithm with some probability and the following *reservation algorithm* with the remaining probability:

- Pick a random value $\tau \in_R \mathsf{Binom}(n, 1/2)$. Observe the first τ agents in the random sequence without assigning any goods. Call these agents the *sample set* S.
- Let I be the set of K agents of largest value in S. For each integer i, suppose there are r_i agents in I having values between 2^{i-1} and 2^i (henceforth called value class i). Let a be the index of the largest non-empty value class. We partition the goods into subsets $T_i = \{1 + \sum_{j=i+1}^{a} r_j, \ldots, r_i + \sum_{j=i+1}^{a} r_j\}$, and we *reserve* the goods in subset T_i for agents in value class i. That is, for each i, we will allocate the r_i goods in subset T_i only to agents in value class i. Note that subset T_a consists of the first (heaviest) r_a goods, subset T_{a-1} consists of the next r_{a-1} goods, and so on.
- Upon observing an agent e in the remaining $n - \tau$ agents, let i be its value class. If there are unassigned goods in subset T_i then assign agent e the heaviest unassigned good in T_i.

The bulk of the work is to prove that the above reservation algorithm is constant-competitive when the optimal solution has many agents in each value class. The intuition for this fact is as follows: if the optimal solution contains many agents from, say, value class i, then about half of them will appear in the sample set. Each such agent will cause the reservation algorithm to reserve one good for agents of value class i. Thus, the remaining half of the agents from value class i in the optimal solution which appear after the sample set will be assigned goods. This, combined with the observation that goods reserved for value class i are at least as heavy as goods assigned to value class i in the optimal solution, will enable us to prove a constant competitive ratio.

We first introduce some notation describing the optimal and algorithmic solutions. Fix an optimal solution, and

let u_i be the number of goods assigned to agents in value class i by this solution. Recall that the reservation algorithm reserves r_i goods for value class i. Let r'_i be the number of goods that the reservation algorithm reserves for value class i because of an agent in I that is also in the optimal solution, so $r'_i \le r_i$ and $r'_i \le u_i$. Let f_i be the number of goods reserved for value class i that the algorithm actually ends up assigning to agents.

We first prove that the reservation algorithm assigns sufficiently many goods to agents in value class i (i.e., that f_i is large).

Lemma 3.1 *If there are at least two agents from value class i that are assigned goods in the optimal solution (i.e., $u_i \ge 2$), then $\mathbf{E}[f_i] \ge \frac{1}{4}u_i$.*

Proof: If the algorithm runs out of goods to give to agents in value class i then $f_i = r_i \ge r'_i$. If the algorithm does not run out of goods to give to agents in value class i then every agent in class i that comes after the sample set gets a good, so certainly the agents from value class i that get a good in the optimal solution but did not appear in the sample get a good, and thus in this case $f_i \ge u_i - r'_i$. So $f_i \ge \min\{r'_i, u_i - r'_i\}$. Since the optimal solution gives goods to the most valuable K agents, any agent that is given a good in the optimal solution and also appears in the algorithm's sample reserves a good for its class, so r'_i is drawn from the binomial distribution with u_i trials and probability 1/2. So if the optimal solution gives goods to at least two agents with values in class i (i.e. $u_i \ge 2$), this implies that $\mathbf{E}[\min\{r'_i, u_i - r'_i\}] \ge \frac{1}{4}u_i$, and thus $\mathbf{E}[f_i] \ge \frac{1}{4}u_i$. So in expectation the reservation algorithm gives goods to at least $\frac{1}{4}$ as many agents from value class i as the optimal solution does. ■

Next we prove that the goods reserved by the algorithm are at least as heavy as the goods used by the optimal solution. Let $b_i = \sum_{j=i+1}^{a} r_j$ and let $o_i = \sum_{j=i+1}^{a} u_j$, so $b_i + 1$ is the "starting good" for class i in the solution given by the algorithm and $o_i + 1$ is the "starting good" for class i in the optimal solution. The proof of the following lemma is immediate and is deferred to the full version.

Lemma 3.2 *$b_i \le o_i$ for all $i \in [a]$*

Let s_O be the optimal assignment, and let s_R be the assignment returned by the reservation algorithm. Let $\mathsf{OPT}_i = \sum_{j=o_i+1}^{o_i+u_i} v(s_O(j))w(j)$ be the contribution of value class i to the optimal solution, so $\mathsf{OPT} = \sum_i \mathsf{OPT}_i$. Similarly, let $R_i = \sum_{j=b_i+1}^{b_i+f_i} v(s_A(j))w(j)$ be the contribution of class i to the solution given by the algorithm. We now claim that as long as at least two agents from value class i get goods in the optimal solution, the algorithm does well in expectation.

Lemma 3.3 *If there are at least two agents from value class i that are assigned goods in the optimal solution (i.e., $u_i \ge 2$), then $\mathbf{E}[R_i] \ge \frac{1}{8}\mathbf{E}[\mathsf{OPT}_i]$*

Proof: We know that the weights of goods are non-increasing with their index, and from Lemma 3.2 we know that the starting good for value class i in the solution returned by the algorithm has index no larger than the starting good for value class i in the optimal solution (and thus is at least as heavy). Lemma 3.1 also implies that in expectation at least 1/4 as many goods are assigned to value class i by the algorithm as in the optimal solution. Putting these together, we get that

$$\begin{aligned}
\mathbf{E}[R_i] &\ge \mathbf{E}\left[2^{i-1}\textstyle\sum_{j=b_i+1}^{b_i+f_i} w(j)\right] \\
&\ge \mathbf{E}\left[2^{i-1}\textstyle\sum_{j=o_i+1}^{o_i+f_i} w(j)\right] && \text{(Lemma 3.2)} \\
&\ge \tfrac{1}{4}2^{i-1}\textstyle\sum_{j=o_i+1}^{o_i+u_i} w(j) && \text{(Lemma 3.1)} \\
&\ge \tfrac{1}{8}2^{i}\textstyle\sum_{j=o_i+1}^{o_i+u_i} w(j) \ge \tfrac{1}{8}\mathsf{OPT}_i.
\end{aligned}$$

proving the lemma. ■

If there are value classes that only have one agent that gets a good in the optimal solution, then the reservation algorithm might not work because if that one agent appears in the sample then it will not appear in the final solution, and if it appears after the sample then there might not be a good reserved for it, so Lemma 3.1 would not hold. We handle this case by essentially running the classical secretary algorithm in parallel as follows.

Theorem 3.4 (Wtd Secretary: Upper Bound) *Let algorithm $\mathcal{A}$ run the reservation algorithm with probability $\frac{8}{3e+8}$, and with the remaining probability $\frac{3e}{3e+8}$, run the classical secretary algorithm assigning the heaviest good to the single agent chosen. Algorithm $\mathcal{A}$ is $8 + 3e$ competitive.*

Proof: Let H be the set of value classes that have at least 2 agents that get goods in the optimal solution and let T denote the rest of the value classes. We denote $OPT_H = \sum_{i\in H} \mathsf{OPT}_i$ and $OPT_T = \sum_{i\in T} \mathsf{OPT}_i$. Clearly $\mathsf{OPT} = OPT_H + OPT_T$. By Lemma 3.3 the reservation algorithm returns a solution with expected value at least $\frac{1}{8}\mathsf{OPT}_H$. Let $v_{max} = \max_{e\in U} v(e)$. It is easy to observe that $OPT_T \le 3 \cdot w(1) \cdot v_{max}$. As the secretary algorithm picks v_{max} with probability $1/e$, if we run the secretary algorithm and assign the heaviest good to the agent chosen we get at least $\frac{1}{3e}\mathsf{OPT}_T$ in expectation. We conclude that the expected value of the solution returned by $\mathcal{A}$ is at least

$$\tfrac{3e}{3e+8} \cdot \tfrac{1}{3e} \cdot \mathsf{OPT}_T + \tfrac{8}{3e+8} \cdot \tfrac{1}{8} \cdot \mathsf{OPT}_H = \tfrac{1}{3e+8}\mathsf{OPT},$$

which completes the proof. ■

Remark: We can slightly improve the above constant using the following optimizations (which will result in more involved analysis of the algorithm). First, we are currently using a scale of powers of 2, this can be replaced by a scale of powers of α for some $\alpha > 1$. Additionally, the scale can be shifted randomly. Finally, the probabilities in which we run each of the algorithms can be tuned to the new scale. The details for which are left to the full version of this paper.

4 Time-dependent Weights, or the Discounted Secretary Problem

In the discounted secretary problem, we are given a function $d(\cdot)$ that maps time instants to "discounts", such that the actual benefit obtained by picking an element e of intrinsic value $v(e)$ at time t is actually the product $v(e) \cdot d(t)$. This is a natural model when picking an item is more valuable at some times rather than others: while the problem has been studied in the simple case of $d(t) = \beta^t$ for some $\beta < 1$, here we consider general *time-varying* functions $d(t)$.

We first show that the problem is fairly hard in general. We show an $\Omega(\log n / \log\log n)$ lower bound on the competitive ratio of any algorithm, and show a nearly matching $O(\log n)$ upper bound. Surprisingly, the problem becomes much easier with a small amount of information about the input. We show that knowing an estimate of $\mathbf{E}[\mathsf{OPT}]$ enables one to design a constant competitive algorithm.

A Warmup Example. Consider the simple case when $d(t) = \beta^t$ for some constant β bounded away from 1. A simple constant-competitive algorithm is one that always picks the first element. This algorithm has an expected value of $\mathbf{E}[\mathsf{ALG}] = \sum_{i=1}^{n} v(i) \cdot \frac{1}{n} \cdot \beta^1$. On the other hand, the expected value that OPT gets from time step j is at most $\sum_{i=1}^{n} v(i) \cdot \frac{1}{n} \cdot \beta^j$, thus the expected value that OPT gets is at most $\sum_{i=1}^{n} v(i) \cdot \frac{1}{n} \cdot \sum_j \beta^j \leq \frac{1}{1-\beta}\mathbf{E}[\mathsf{ALG}]$. As β is a fixed constant this is constant-competitive (even though, as noted in [RP76], $\mathsf{OPT} \to 0$ as $n \to \infty$).

4.1 Discounted Secretary: General Case

It turns out the the discounted secretary problem is significantly harder than other variants of the secretary problem, and we show a lower bound of $\Omega(\frac{\log n}{\log\log n})$ on the competitive ratio in this case. We also give an algorithm with an almost-matching competitive ratio of $O(\log n)$.

4.1.1 A Nearly-Logarithmic Lower Bound

We construct a discount function d and a family of instances $\mathcal{I}_1, \ldots, \mathcal{I}_{2c}$ such that no randomized algorithm can be $\Theta(\frac{\log n}{\log\log n})$-competitive on all the $\mathcal{I}_t$'s. More formally, we use the following definitions:

- **Instance Size:** Let $L = c$ be an integer and let $n = L^{4c}$. Thus $L = c = \Theta(\frac{\log n}{\log\log n})$.
- **Discount Function:** For $t \leq 2c$, let $n_t = L^{2t}$. For the discount function, we use a step function $d(j) = L^{-1}$ for $1 \leq j \leq n_1$, and $d(j) = L^{-t}$ for $n_{t-1} < j \leq n_t$.
- **Instances:** Let K be a large enough integer (say n^2). The instance $\mathcal{I}_1$ consists of $\frac{n}{n_1}$ K's and the remaining zeroes. For $t < 2c$, $\mathcal{I}_{t+1}$ is obtained inductively from $\mathcal{I}_t$ by changing n_t/n_{t+1} of the K^t's to K^{t+1}. Thus $\mathcal{I}_t$ has $\frac{n}{n_t}$ K^t's.

Lemma 4.1 (Estimate on $\mathbf{E}[\mathsf{OPT}(\mathcal{I}_t)]$) *For $t \leq 2c$, $OPT(\mathcal{I}_t)$ is at least $(1 - \frac{1}{e})K^t L^{-t}$*

Proof: With probability at least $(1 - \frac{1}{e})$ over the random permutation, one of the n/n_t K^t values falls in the first n_t slots, leading to a value of at least $K^t L^{-t}$. ■

Let $\mathcal{A}$ be a $c/10$-competitive algorithm. We argue that $\mathcal{A}$ must have a large probability of picking in item in each of the intervals (n_t, n_{t+1}).

Lemma 4.2 *Let $\mathcal{A}$ be a $c/10$-competitive algorithm. Then on instance $\mathcal{I}_t$, the probability that $\mathcal{A}$ picks one of the first n_t items is at least t/c.*

Proof: We prove the claim by induction on t. For the base case, note that the most $\mathcal{A}$ can get from time steps $n_1 + 1$ onwards is $K/L^2 = \frac{1}{c}(\frac{K}{L})$. Thus to be $\frac{c}{10}$ competitive, it must in expectation get value at least $\frac{2}{c}(\frac{K}{L})$ from the first n_1 time steps. Since $\mathcal{A}$ never gets more than K/L on $\mathcal{I}_1$, the base case holds.

Assume that the claim holds for $\mathcal{I}_t$. Consider a run of the algorithm on instance $\mathcal{I}_{t+1}$. Note that $\mathcal{I}_{t+1}$ differs from $\mathcal{I}_t$ only in $\frac{n}{n_{t+1}}$ of the items. The probability that any one of these items occurs in the first n_t time steps is at most $\frac{n_t}{n_{t+1}}$. Thus except with probability $\frac{n_t}{n_{t+1}} = \frac{1}{L^2}$, the behavior of $\mathcal{A}$ on $\mathcal{I}_t$ and $\mathcal{I}_{t+1}$ is indistinguishable in the first n_t steps. Thus by the induction hypothesis, the algorithm accepts an item by time n_t with probability at least $(1 - \frac{1}{L^2})(t/c)$.

The expected revenue of $\mathcal{A}$ in the first n_t time steps, on instance $\mathcal{I}_{t+1}$ is bounded by $K^{t+1}(L^{-t}\frac{n_t}{n_{t+1}} + L^{-(t-1)}\frac{n_{t-1}}{n_{t+1}} + \ldots + L^{-1}\frac{n_1}{n_{t+1}}) + K^t L^{-1}$, where the first term bounds the expected contribution from the K^{t+1}'s, and the second term bounds the most one can get from the smaller items. Since this is at most $4OPT/c$, $\mathcal{A}$ must get at least $6OPT/c$ from time steps $n_t + 1$ onwards. As in the base case, time

steps $n_{t+1}+1$ and higher cannot contribute more than $2OPT/c$, so that the algorithm must get contribution $4OPT/c$ from time steps $[n_t+1, n_{t+1}]$. Thus the probability that the algorithm picks an item in time steps $[n_t+1, n_{t+1}]$ is at least $\frac{2}{c}$. Since $\mathcal{A}$ picks exactly one item, this event is disjoint from $\mathcal{A}$ picking an item in time steps $[1, n_t]$. Thus the probability that $\mathcal{A}$, on instance $\mathcal{I}_{t+1}$ picks an item in times steps $[1, n_{t+1}]$ is at least $(1-\frac{1}{L^2})(t/c)+(2/c) \geq (t+1)/c$. The claim follows. ■

Now we can prove the main lower bound theorem.

Theorem 4.3 (Discounted Sec'y: Lower Bound) *Every algorithm $\mathcal{A}$ for the discounted secretary problem has $\mathbf{E}[\mathsf{OPT}]/\mathbf{E}[\mathcal{A}] \geq \Omega(\log n/\log\log n)$ in the worst case.*

Proof: If $\mathcal{A}$ is $c/10$-competitive, then Lemma 4.2 implies that on instance $\mathcal{I}_{2c}$, the probability that $\mathcal{A}$ picks an item in the first n_{2c} time steps is larger than 1, which is a contradiction. Thus $\mathcal{A}$ cannot be $c/10$-competitive. ■

4.1.2 A Logarithmic Upper Bound

Theorem 4.4 (Discounted Sec'y: Upper Bound) *There is an algorithm $\mathcal{A}$ for the discounted secretary problem such that $\mathbf{E}[\mathsf{OPT}]/\mathbf{E}[\mathcal{A}] \leq O(\log n)$.*

Proof: Let d_{max} be the maximum discount and let v_{max} be the maximum value. For $c \geq 1$ let $I_c = (2^{-c}d_{max}, 2^{-(c-1)}d_{max}]$ be the interval defining the c-th discount class, let $P_c = \{i \in [n] : d(i) \in I_c\}$ be the set of times that have discount value in class c. Thus $\mathsf{OPT} = \sum_c \sum_{i\in P_c} d(i) \sum_j v(j) Pr[\pi(i)=j \wedge d(i)v(j) \in \mathsf{OPT}]$. Let OPT_c denote the term corresponding to P_c so that $\mathsf{OPT} = \sum_c \mathsf{OPT}_c$. Clearly $\mathsf{OPT}_1 \geq v_{max}d_{max}/n$, since v_{max} occurs in the location corresponding to d_{max} with probability at least $\frac{1}{n}$. Moreover, OPT_c equals

$$\begin{aligned}\sum_{i\in P_c} d(i) & \sum_j v(j) Pr[\pi(i)=j \wedge d(i)v(j) \in \mathsf{OPT}] \\ &\leq \sum_{i\in P_c} 2^{-(c-1)}d_{max} \sum_j v(j) \\ &\leq 2^{-(c-1)}d_{max}|P_c|nv_{max}\end{aligned}$$

since each $v(j)$ is bounded by v_{max}. Moreover, since $|P_c|$ is at most n, we conclude that $\mathsf{OPT}_c \leq 2^{-c}2n^2 d_{max}v_{max}$. It follows that $\sum_{c\geq 3\log n+2} \mathsf{OPT}_c \leq \mathsf{OPT}/2$, so that $\sum_{c=1}^{3\log n+1} \mathsf{OPT}_c \geq \mathsf{OPT}/2$. Thus OPT gets most of its value for the top $O(\log n)$ discount scales.

Our algorithm $\mathcal{A}$ chooses a $c \in [3\log n+2]$ uniformly at random, and then runs the classical secretary algorithm in the time steps $i \in P_c$. Clearly the expected value that the offline optimum gets from time steps in P_c equals OPT_c. Thus the expected value of the offline optimum for this restricted problem is at least as large, and the classical secretary algorithm gets an expected value $\mathsf{OPT}_c/2e$ (we lose an additional factor of two since we treat the discount values in P_c equally). The claim follows. ■

When there are both weights and discounts, we can combine Theorem 4.4 with Theorem 3.4 to get the following result. We defer the proof to the full version.

Theorem 4.5 *There is an algorithm for the weighted discounted secretary problem that is $O(\log n)$-competitive*

Proof Sketch: In the unweighted case the proof of Theorem 4.4 implies that we can ignore all but the first $\Theta(\log n)$ discount classes. Using the same reasoning, in the weighted case we can ignore all but the first $\Theta(\log(nK))$ discount classes, where the extra K is due to the ability to assign K goods. Now we can just choose a $c \in [\Theta(\log(nK))]$ uniformly at random and run the weighted secretary algorithm of Theorem 3.4 in the time steps with discount in class c. Following the analysis of Theorem 4.4, this is an $O(\log(nK))$-competitive algorithms, and thus is also $O(\log n)$-competitive. ■

4.2 Discounted Secretary with Known OPT

In this section we consider the case when the algorithm knows a good estimate for $\mathbf{E}[\mathsf{OPT}]$ ahead of time. Interestingly, we can show that even if we know *all the values* and not just $\mathbf{E}[\mathsf{OPT}]$ in advance, no online algorithm can be perfectly competitive. This is in contrast to the case without discounts (i.e., $d(t) = 1$), in which the naïve algorithm that knows all the values can pick a element of maximal value. (Proof in Appendix A.1.)

Theorem 4.6 (Known-OPT: Lower Bound) *For any $\epsilon > 0$, every algorithm $\mathcal{A}$ for the discounted secretary problem has $\mathbf{E}[\mathsf{OPT}]/\mathbf{E}[\mathcal{A}] \geq \sqrt{2}-\epsilon$, even when $\mathcal{A}$ knows the set of values (and hence $\mathbf{E}[\mathsf{OPT}]$) in advance.*

The main result of this section shows that one can get a constant competitive algorithm if we have a good estimate Z of $\mathbf{E}[\mathsf{OPT}] = \mathbf{E}_\pi[\max_{t\in[n]} d(t)\cdot v(\pi(t))]$.

> **Algorithm $\mathcal{A}$:** Suppose $Z \leq \mathbf{E}[\mathsf{OPT}]$. Pick the first element e seen (say, at time j) that satisfies $v(i)d(j) \geq Z/2$.

Theorem 4.7 (Known-OPT: Upper Bound) *If $Z \leq \mathbf{E}[\mathsf{OPT}]$, the algorithm $\mathcal{A}$ for the discounted secretary problem satisfies $\mathbf{E}[\mathcal{A}] \geq Z/4$.*

Proof: The proof considers two cases. In the first (easier) case, suppose the algorithm $\mathcal{A}$ picks *some* element with probability at least $1/2$. Then with probability at least $1/2$, it gets benefit at least $Z/2$, and hence its expected performance is at least $Z/4$.

In the second case, suppose the algorithm $\mathcal{A}$ does not pick an element with probability at least $1/2$: i.e., we are in the case where at least half the permutations cause all products $\{v(e)\, d(\pi(e))\}_{e \in U}$ to be small (and hence are "rejecting"). In this case, we can show that OPT gets most of its benefit from the other, "accepting" permutations. Moreover, on these accepting permutations, we would pick an element differently from OPT only when there was some 'blocking' element with $v(e)\, d(\pi(e)) \geq Z/2$: but for the same reason that there are few accepting permutations, there are few accepting permutations with such a blocking element. Hence we pick the same elements OPT picks on the "accepting" permutations with high probability, and have a good performance.

Let us now make this general idea for the second case precise. Let us write

$$\mathbf{E}[\mathsf{OPT}] = \sum_{\pi \in S_n} \frac{1}{n!} \max_{i=1}^{n} \{d(i)v(\pi(i))\}$$

Let $S_{acc} = \{\pi \in \mathfrak{S}_n : \max_{i=1}^{n}\{d(i)v(\pi(i)) \geq Z/2\}$ be the "accepting permutations" on which the algorithm picks some element. The contribution to $\mathbf{E}[\mathsf{OPT}]$ from these accepting permutations is

$$\begin{aligned} L &\stackrel{def}{=} \textstyle\sum_{\pi \in S_{acc}} \frac{1}{n!} \max_{i=1}^{n}\{d(i)v(\pi(i))\} \\ &= \mathbf{E}[\mathsf{OPT}] - \textstyle\sum_{\pi \notin S_{acc}} \frac{1}{n!} \max_{i=1}^{n}\{d(i)v(\pi(i))\} \\ &\geq \mathbf{E}[\mathsf{OPT}] - \textstyle\sum_{\pi \notin S_{acc}} \frac{1}{n!}(Z/2) \\ &\geq \mathbf{E}[\mathsf{OPT}] - \tfrac{Z}{2} \geq \tfrac{Z}{2}, \end{aligned} \tag{4.1}$$

where we used the fact that $Z \leq \mathbf{E}[\mathsf{OPT}]$. To complete the proof, it suffices to show that $\mathcal{A}$ has expected value at least $L/2$, which is at least $Z/4$ by (4.1). We begin by rewriting L as

$$\begin{aligned} L = \textstyle\sum_{i=1}^{n} \sum_{j : d(i)v(j) \geq \frac{Z}{2}} \frac{1}{n} \cdot d(i)v(j) \\ \times \mathbf{Pr}[d(i)v(j) \text{ highest } |\pi(i) = j] \end{aligned} \tag{4.2}$$

(Here we implicitly assume that there is exactly one product $d(i)v(j)$ which is highest: we break ties in some consistent fashion.) Combining (4.1) and (4.2) gives us

$$\textstyle\sum_{i=1}^{n} \sum_{j : d(i)v(j) \geq \frac{Z}{2}} \frac{1}{n} \cdot d(i)v(j) \geq \frac{Z}{2} \tag{4.3}$$

For each such i, j pair where $d(i)v(j) \geq Z/2$, let G_{ij} be the set of permutations on which $\mathcal{A}$ chooses element j at time i, getting benefit $d(i)v(j)$: let us call these "good". (Note that, in these permutations, $d(i)v(j)$ must be the *first* product which is at least $Z/2$.) The performance of the algorithm is

$$\mathbf{E}[\mathcal{A}] = \textstyle\sum_{i=1}^{n} \sum_{j : d(i)v(j) \geq \frac{Z}{2}} d(i)v(j) \cdot \frac{|G_{ij}|}{|\mathfrak{S}_n|} \tag{4.4}$$

The following claim shows there are many "good" permutations.

Claim 4.8 *For each i, j such that $d(i)v(j) \geq Z/2$, $|G_{ij}| \geq |\mathfrak{S}_n \setminus S_{acc}|/n \geq |\mathfrak{S}_n|/2n$.*

Proof: We show the first inequality by giving an n-to-1 map f_{ij} from $\mathfrak{S}_n \setminus S_{acc}$ to G_{ij}, the second inequality follows from the fact that at most half the permutations are in S_{acc}.

Define $f_{ij}(\pi)$ by changing π and mapping element j to location i, swapping it with whatever was there originally. In other words, if $i' = \pi^{-1}(j)$, let $f_{ij}(\pi)(i) = \pi(i') = j$, let $f_{ij}(\pi)(i') = \pi(i)$, and for $k \neq i, i'$ let $f_{ij}(\pi)(k) = \pi(k)$. Showing f_{ij} is a n-to-1 map is straight-forward; to show that $\pi' = f_{ij}(\pi)$ is in G_{ij}, we need to show that $d(i)v(\pi'(i)) = d(i)v(j) \geq Z/2$ (which follows from our choice of i, j), and that no other position i' has $d(i')v(\pi'(i')) \geq Z/2$ (which follows almost as easily from the fact that $\pi \notin S_{acc}$ and hence did not have any positions i' such that $d(i')v(\pi(i')) \geq Z/2$. ∎

The rest of the proof of Theorem 4.7 is immediate: by (4.4) and Claim 4.8, it follows that

$$\mathbf{E}[\mathcal{A}] \geq \textstyle\sum_{i=1}^{n} \sum_{j : d(i)v(j) \geq \frac{Z}{2}} d(i)v(j) \cdot \frac{1}{2n} \geq Z/4,$$

where the last step is by equation (4.3). ∎

To use Theorem 4.7 fruitfully and obtain benefit $\approx \mathbf{E}[\mathsf{OPT}]/4$, the bound Z should be close to $\mathbf{E}[\mathsf{OPT}]$: such an estimate could come from expert knowledge, prior runs of the problem, or some other source. A special case when such an estimate of $\mathbf{E}[\mathsf{OPT}]$ can be obtained is the situation when we know the values of all the elements. In this case, we can use a sampling-based estimator for $\mathbf{E}[\mathsf{OPT}]$ (as shown in Lemma A.1) to get $\mathbf{E}[\mathcal{A}] \geq \frac{(1-\delta)(1-\epsilon)}{4} \mathbf{E}[\mathsf{OPT}]$.

5 Extensions to Matroid Secretary Problems

In this section, we show how the secretary problem on many classes of matroids can be "reduced" to partition matroids. Moreover, we extend our results for the discounted and weighted secretary problems to partition matroids, and hence to such classes of matroids as well. This reduction to partition matroids also gives a simple unified view of several results of Babaioff et al. [BIK07], and also improves some of the results from their paper.

Recall that a matroid $\mathcal{M} = (U, \mathcal{I})$ consists of a ground set U, and a collection $\mathcal{I}$ of subsets of U that is

closed under taking subsets satisfying the well-known exchange conditions. A *partition matroid* $(U, \mathcal{I})$ is one where there is some partition P of U, and $S \in \mathcal{I}$ if and only if I has at most one element from each set in P. The following definition formalizes the notion of a "reduction" from arbitrary matroids to partition matroids.

Definition 5.1 (An α-Partition Property) *A matroid $\mathcal{M} = (U, \mathcal{I})$* satisfies an α-partition property *if one can (randomly) define a* partition matroid $\mathcal{M}' = (U', \mathcal{I}')$ *on some subset U' of the universe U such that for any values of the elements U, we have*

- *E(value of max-weight base in $\mathcal{M}'$) $\geq 1/\alpha \times$ value of max-weight base in $\mathcal{M}$.*
- *Every independent set in $\mathcal{M}'$ is an independent set in M.*

In a *graphic matroid* the ground set is the set of edges of some graph $G = (V, E)$, and $S \subseteq E$ is independent if and only if it does not contain a cycle. In a *uniform matroid* the independent sets are all subsets of size at most some k. Finally, in a *transversal matroid* the ground set is the set of left vertices from some bipartite graph $G = (L, R, E)$, and a set $S \subseteq L$ is independent if and only if there is a perfect matching of S to nodes in R. All of these matroids satisfy an α-partition property:

Theorem 5.2 *An α-partition property can be shown for the following classes of matroids:*

- *Any graphical matroid satisfies a 3-partition property.*
- *Any uniform matroid satisfies a $e/(e-1)$-partition property.*
- *Transversal matroids satisfy a d-partition property, where d is the maximum degree of vertices on the left, as well as a $4k/d$-partition property, where k is the rank of the matroid.*

The proofs for these reductions appear in Appendix B; as an example, we give the reduction for graphical matroids.

Lemma 5.3 (Graphical Matroids) *Any graphical matroid satisfies a 3-partition property.*

Proof: Let G be a graph defining a graphic matroid. Pick an edge $\{u, v\}$ from G uniformly at random, with probability $\frac{1}{2}$ color u red and v blue, and with probability $\frac{1}{2}$ color u blue and v red. Then independently color every other node red or blue, each with probability $\frac{1}{2}$. Create a part in the partition matroid for each red node x, and add to it all the red-blue edges incident on x. Then run this procedure recursively on the graph induced by the edges that have both endpoints colored blue to create more sets in the partition.

It is easy to see that picking one bichromatic edge incident on each red node will result in a forest. It is also clear that taking the union of such a forest with any set of blue-blue edges that is itself a forest still gives a forest, implying that any set which is independent in the partition matroid we create is independent in the original graphic matroid.

For a graph G, let $\mathsf{OPT}(G)$ be the value of the optimal independent set in the graphic matroid defined by G. Let $v(G)$ be a random variable denoting the value of the maximum independent set in the partition matroid constructed by this reduction. We claim that $\mathbf{E}[v(G)] \geq \frac{1}{3}\mathsf{OPT}(G)$, and prove it by induction on the number of edges of G. For the base case, if G only has one edge then we color it bichromatically with probability 1, so $\mathbf{E}[v(G)] = \mathsf{OPT}(G) \geq \frac{1}{3}\mathsf{OPT}(G)$ as claimed.

For the inductive step, let X_{rb} be a random variable denoting the value of the optimal independent set from the partition matroid corresponding just to the sets we created for the red nodes. Let T be the optimal forest (which without loss of generality is a tree, since otherwise we just analyze each component separately). Root T arbitrarily. After the random coloring, the set of edges that go from a red node to a blue parent are clearly bichromatic and will get assigned to different sets in the partition corresponding to red nodes, so the optimal independent set in the partition matroid is at least as large as their sum. Every edge in T has probability $1/m$ (where m is the number of edges in G) of being the initial edge chosen to be colored bichromatically, and if it is the one chosen then with probability $1/2$ the parent is blue and the child is red. If it is not chosen then it still has probability $1/4$ of having its parent colored blue and its child red. So the total probability that it is colored in this way is at least $\frac{1}{2m} + (1 - \frac{1}{m})\frac{1}{4} = \frac{1}{4} + \frac{1}{4m}$, so by linearity of expectations $\mathbf{E}[X_{rb}] \geq (\frac{1}{4} + \frac{1}{4m})\mathsf{OPT}(G)$.

Clearly $v(G) = X_{rb} + v(G_{bb})$, where G_{bb} is the graph induced by edges that have both endpoints colored blue. The probability that an edge in T is colored monochromatically blue is at least $(1-\frac{1}{m})\frac{1}{4} = \frac{1}{4} - \frac{1}{4m}$, since if it is not picked as the initial bichromatic edge its endpoints are both colored blue with probability $1/4$. By linearity of expectations this means that $\mathbf{E}[\mathsf{OPT}(G_{bb})] \geq (\frac{1}{4} - \frac{1}{4m})\mathsf{OPT}(G)$. Also, since we colored at least one edge bichromatically by induction we know that $\mathbf{E}[v(G_{bb})] \geq \frac{1}{3}\mathsf{OPT}(G_{bb})$. So $\mathbf{E}[v(G)] = \mathbf{E}[X_{rb} + v(G_{bb})] = \mathbf{E}[X_{rb}] + \mathbf{E}[v(G_{bb})] \geq (\frac{1}{4} + \frac{1}{4m})\mathsf{OPT}(G) + \mathbf{E}\left[\frac{1}{3}\mathsf{OPT}(G_{bb})\right] \geq (\frac{1}{4} + \frac{1}{4m})\mathsf{OPT}(G) + \frac{1}{3}(\frac{1}{4} - \frac{1}{4m})\mathsf{OPT}(G) \geq \frac{1}{3}\mathsf{OPT}(G)$ ■

The reductions of Theorem 5.2 motivate the study of the matroid secretary problem, the weighted version, and the discounted version on partition matroids.

Theorem 5.4 *On partition matroids, there is an e-competitive algorithm for the matroid secretary problem,*

a $(16+3e)$-competitive algorithm for the weighted matroid secretary problem, an $O(\log n)$-competitive algorithm for the discounted matroid secretary problem, and a $(\frac{(1-\delta)(1-\epsilon)}{4})$-competitive algorithm for the discounted matroid secretary problem if we know all of the values.

Proof: The e-competitive algorithm for the matroid secretary problem is trivial: just run the classical secretary algorithm on each set in the partition. The same argument combined with Theorem 4.4 gives the theorem for the discounted case, and combined with Theorem 4.7 and Lemma A.1 gives the theorem for the discounted case with known-OPT (we defer the details to the full version). The proof for the weighted problem is given in Appendix C. ■

This theorem, combined with the definition of the α-partition property, immediately implies that if a matroid satisfies an α-partition property, then there is an $(e\alpha)$-competitive algorithm for the matroid secretary problem, a $((16+3e)\alpha)$-competitive algorithm for the weighted secretary problem, an $O(\alpha \log n)$-competitive algorithm for the discounted matroid secretary problem, and a $(\frac{(1-\delta)(1-\epsilon)}{4}\alpha)$-competitive algorithm for the discounted matroid secretary problem if we know all of the values. Thus Theorem 5.2 implies that there are constant-competitive algorithms for graphic, uniform, and low- and high-degree transversal matroids in all variants but the general discounted setting, in which case there are $O(\log n)$-competitive algorithms.

If we do not know that the matroid we are working with satisfies an α-partition property, we extend Theorem 4.4 to give the following bound for the discounted problem. We defer the proof to Appendix D.

Theorem 5.5 *There is an algorithm $\mathcal{A}$ for the discounted secretary problem on matroids such that $\mathbf{E}[\mathsf{OPT}]/\mathbf{E}[\mathcal{A}] \leq O(\log k \cdot \log n)$ where k is the rank of the matroid $\mathcal{M}$ which is the input to the algorithm.*

In Section 4.2 we saw that if one knows all the values, or knows (a good estimate of) the expectation of OPT this is enough to break the lower bound and achieve a constant competitive algorithm. For discounted secretary problem on matroids knowing the maximal value is also very helpful. In such case we can improve the result of Theorem 5.5.

Theorem 5.6 *There is an algorithm $\mathcal{A}$ for the discounted secretary problem on matroids when the maximal value is known such that $\mathbf{E}[\mathsf{OPT}]/\mathbf{E}[\mathcal{A}] \leq O(\log n)$.*

Proof: For $i \in Z$, let scale P_c be the set of all pairs i,j such that $d(i)v(j) \in (2^{c-1}, 2^c]$. Thus $OPT \leq 2^c \sum_{(i,j)\in P_c} Pr[\pi(i)=j \wedge d(i)v(j) \in OPT]$. Since v_{max} falls in the maximum discount location with probability $\frac{1}{n}$, $OPT \geq v_{max}d_{max}/n$. Let $c_{max} = \lceil \log v_{max}d_{max} \rceil$ be the index of the highest non-empty scale. Since $|P_c| < kn$, it follows that that contribution of level c to OPT, $2^c \sum_{(i,j)\in P_c} Pr[\pi(i)=j \wedge d(i)v(j) \in OPT]$ is bounded above by $nk2^c$. Thus for $c_{min} = c_{max} - \lceil \log n^2 k \rceil - 2$, the total contribution to $E[OPT]$ for scales c_{min} and below is at most $OPT/2$.

The algorithm $\mathcal{A}$ samples a random integer c in $[c_{min}+1, c_{max}]$ and runs the greedy algorithm, restricted to elements at scale c or higher. The expected contribution of this scale to OPT is $\Omega(\frac{OPT}{\log n})$. It is easy to check that this gives an $O(\log n)$-competitive ratio. ■

References

[BIK07] Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In *SODA '07*, pages 434–443, 2007.

[BIKK07] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg. A knapsack secretary problem with applications. In *APPROX '07*, 2007.

[DLR71] Cyrus Derman, Gerald J. Lieberman, and Sheldon M. Ross. A sequential stochastic assignment problem. *Management Sci.*, 18:349–355, 1971.

[Dyn63] E. B. Dynkin. Optimal choice of the stopping moment of a Markov process. *Dokl. Akad. Nauk SSSR*, 150:238–240, 1963.

[Fer89] T.S. Ferguson. Who solved the secretary problem? *Statistical Science*, 4:282–296, 1989.

[Fre83] P. R. Freeman. The secretary problem and its extensions: a review. *Internat. Statist. Rev.*, 51(2):189–206, 1983.

[GM07] A. Gershkov and B. Moldovanu. The dynamic assignment of heterogenous objects: A mechanism design approach. Technical report, University of Bonn, 2007.

[HKP04] Mohammad Taghi Hajiaghayi, Robert Kleinberg, and David C. Parkes. Adaptive limited-supply online auctions. In *EC '04*, pages 71–80, 2004.

[Kle05] Robert Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *16th SODA*, pages 630–631. ACM, 2005.

[Lin61] D.V. Lindley. Dynamic programming and decision theory. *Applied Statistics*, 10(1):39–51, March 1961.

[LN00] Ron Lavi and Noam Nisan. Competitive analysis of incentive compatible on-line auctions. In *EC '00*, pages 233–241, 2000.

[LN05] Ron Lavi and Noam Nisan. Online ascending auctions for gradually expiring items. In *SODA '05*, pages 1146–1155, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[MMP08] M. Mahdian, P. McAffee, and D. Pennock. The secretary problem with durable employment. personal communication, 2008.

[RP76] Willis T. Rasmussen and Stanley R. Pliska. Choosing the maximum from a sequence with a discount function. *Appl. Math. Optim.*, 2(3):279–289, 1975/76.

A Discounted Secretary Problems: Missing Proofs

A.1 Lower Bound on the Competitive Ratio

Recall that we wanted to show that any algorithm $\mathcal{A}$ for the Discounted Secretary problem has a competitive ratio of no better than $\sqrt{2}-\epsilon$ even when $\mathcal{A}$ knows the set of values.

Proof of Theorem 4.6. Let $c=\sqrt{2}-1$. Let $t=\lfloor n(1-c)\rfloor$, and note that $nc\leq n-t\leq nc+1$. Assume that $v(1)=v(2)=1$ and that $v(i)=0$ for any $i\geq 3$. Let $d(i)=1$ for $i<t$, $d(t)=c\cdot n+1$ and $d(i)=0$ for any $i>t$. The expected value that OPT gets is $\mathbf{E}[\mathsf{OPT}]=1*(1-(\frac{n-t}{n})^2)+c\cdot n\cdot\frac{2}{n}\geq 1-(\frac{nc+1}{n})^2+2c=1+2c-c^2-\frac{2cn+1}{n^2}=4(\sqrt{2}-1)-\frac{2(\sqrt{2}-1)n+1}{n^2}$.

We next show that if $\mathcal{A}$ observes one of the two valuable elements (with value 1) before time t it immediately picks it (this is the optimal online algorithm for this input). This is true as the expected value from picking the element is 1. On the other hand, if there are still k element to come, the expected value from skipping the current element and picking the next valuable element is $1\cdot\frac{k-(n-t)}{k}+cn\cdot\frac{1}{k}\leq\frac{k-cn}{k}+cn\cdot\frac{1}{k}=1$. Thus the algorithm is always (weakly) better off picking the first valuable element it sees.

Now, the expected value that $\mathcal{A}$ gets is $\mathbf{E}[\mathcal{A}]=1*(1-(\frac{n-t}{n})^2)+cn\cdot 2\cdot\frac{n-t}{n}\cdot\frac{1}{n}\leq 1-(\frac{cn}{n})^2+2c\frac{cn+1}{n}=1+c^2+\frac{2c}{n}=4-2\sqrt{2}+\frac{2(\sqrt{2}-1)}{n}$. We conclude that

$$\mathbf{E}[\mathsf{OPT}]/\mathbf{E}[\mathcal{A}]\geq\frac{4(\sqrt{2}-1)-\frac{2(\sqrt{2}-1)n+1}{n^2}}{4-2\sqrt{2}+\frac{2(\sqrt{2}-1)}{n}}$$

As this expression clearly tends to $\frac{4(\sqrt{2}-1)}{4-2\sqrt{2}}=\sqrt{2}$ as n goes to infinity, the claim follows. □

A.2 A Sampling Lemma

The following lemma shows that given access to the element values, one can estimate the expected optimum for the discounted secretary fairly efficiently.

Lemma A.1 *Given access to the element values and an $\epsilon>0$, one can obtain an estimate $\widehat{Z}\in((1-\epsilon)\mathbf{E}[\mathsf{OPT}],(1+\epsilon)\mathbf{E}[OPT])$ with probability $1-\delta$ in time* $\mathrm{poly}(n,\epsilon^{-1},\log\delta^{-1})$.

Proof: For a parameter M to be specified later, sample m permutations $\pi_1,\pi_2,\ldots,\pi_M$ independently from $\mathfrak{S}_n$ and define

$$\widehat{Z}=\frac{1}{M}\sum_t\max_i\{d(i)v(\pi_t(i))\}.$$

Clearly, $\mathbf{E}[\widehat{Z}]=\mathbf{E}[\mathsf{OPT}]$, and it just remains to show that $\widehat{Z}$ is tightly concentrated about its mean. Let $v_{\max}$ denote the value of the most valuable element, and let $d_{\max}$ denote the largest discount. For a given trial t, let $Y_t=\frac{\max_i\{d(i)v(\pi_t(i))\}}{v_{\max}d_{\max}}$, and let $Y=\frac{1}{M}\sum_{i=1}^M Y_i$. (Hence $Y_i\in[0,1]$, $Y=\widehat{Z}/v_{\max}d_{\max}$.)

A quick observation: if we pick the element that appears in the location with the highest discount, the expected benefit is $d_{\max}v_{\max}/n$, and hence $\mathbf{E}[\mathsf{OPT}]\geq d_{\max}v_{\max}/n$, which in turn implies that $\mathbf{E}[Y]\geq\mathbf{E}[\widehat{Z}]/v_{\max}d_{\max}\geq 1/n$. By standard Chernoff bounds, it follows that for $\epsilon\leq 1$,

$$\begin{aligned}\mathbf{Pr}[|Y-\mathbf{E}[Y]|\geq\epsilon\mathbf{E}[Y]] &\leq 2\exp\{-\frac{\epsilon^2\cdot M\mathbf{E}[Y]}{2+\epsilon}\}\\ &\leq 2\exp(-\frac{M\epsilon^2}{3n}),\end{aligned}$$

and thus if we set $M\geq\frac{3}{\epsilon^2}n\ln\frac{2}{\delta}$, this probability is at most δ, which proves the lemma. ■

Note that $\frac{\widehat{Z}}{1+\epsilon}$ is a lower bound for $\mathbf{E}[\mathsf{OPT}]$ with probability $1-\delta$, and hence when the values of the elements are known, we can use Theorem 4.7 to get $\frac{(1-\delta)(1-\epsilon)}{4}\mathbf{E}[\mathsf{OPT}]$.

B Reductions for Several Matroid Classes

Lemma B.1 (Graphical Matroids) *Any graphical matroid satisfies a 3-partition property.*

Lemma B.2 (Uniform Matroids) *Any uniform matroid satisfies a $e/(e-1)$-partition property.*

Proof: Create k bins (where k is the rank of the uniform matroid), and place each element in U into one of these bins uniformly at random. The i-th largest value $v(i)$ is not in the same bin as any larger value with probability at least $(1-1/k)^{i-1}$. This implies that the expected value of the maximal weight basis of the partition matroid is at least

$$\begin{aligned}\textstyle\sum_{i=1}^k v(i)\cdot(1-\frac{1}{k})^{i-1} &\geq \textstyle\sum_{i=1}^k v(i)\cdot\frac{1}{k}\sum_{j=1}^k(1-\frac{1}{k})^{j-1}\\ &= \textstyle\frac{1}{k}\cdot\frac{1-(1-1/k)^k}{1-(1-1/k)}\cdot\sum_{1=1}^n v(i)\\ &= \textstyle(1-(1-1/k)^k)\cdot\sum_{1=1}^n v(i)\\ &\geq \textstyle(1-1/e)\cdot\sum_{1=1}^n v(i)\end{aligned}$$

■

Lemma B.3 (Low-Degree Transversal Matroids) *Transversal matroids satisfy a d-partition property, where d is the maximum degree of vertices on the left.*

Proof: Create a partition in the partition matroid for each vertex on the right, and put a vertex on the left

(i.e., an element of U) in one of the bins corresponding to its neighbors uniformly at random.

Given a maximum independent set I in the transversal matroid, there is some matching between I and the vertices on the right. Clearly each node x in I gets put in the bin corresponding to its neighbor y in this matching with probability at least $1/d$. If this even does happen, then the maximum independent set in the partition matroid contains either x or an element of greater value. Thus the expected value of the maximum independent set is at least $\sum_{x\in I} \frac{1}{d} v(x) = \frac{1}{d}\sum_{x\in I} v(x) = \frac{1}{d}\mathsf{OPT}$ ■

Lemma B.4 (High-Degree Transversal Matroids) *Any transversal matroid satisfies a $4k/d$-partition property, where d is the minimum degree of vertices on the left and k is the rank of the matroid.*

Proof: We use the same reduction as for low-degree transversal matroids. Without loss of generality, let $v(1) \geq v(2) \geq \cdots \geq v(n)$. Then the probability that element i is the most valuable in the set that it is assigned to is at least $1 - \frac{i-1}{d}$, since at most $i-1$ of its neighbors can contain more valuable elements and it has at least d neighbors. Considering only the elements $\{1, 2, \ldots, d/2\}$ and using linearity of expectations, we get that the expected value of the max base in the partition matroid is at least

$$\begin{aligned}\sum_{i=1}^{d/2}\left(1-\tfrac{i-1}{d}\right)v(i) &\geq \frac{1}{2}\sum_{i=1}^{d/2} v(i)\\ &\geq \frac{d}{4k}\sum_{i=1}^{k} v(i) \geq \frac{d}{4k}\mathsf{OPT}\end{aligned}$$

■

C The Weighted Matroid Secretary Problem

In this section, we extend our algorithm for the Weighted Secretary problem to its matroid version. We first give an algorithm and analysis for partition matroids, and then extend this to a wider class of matroids using our general reduction theorems. In this setting we refer to items as seats, and we try to assign elements to seats. The matroid condition is just that the set of elements which get assigned a seat must be independent in the matroid in which we are working.

Let $\mathcal{M} = (U, \mathcal{I})$ be a partition matroid of rank $r(\mathcal{M})$ with partition $\mathcal{P}$. Consider the following *matroid reservation algorithm*. First, sample the beginning using Kleinberg's trick to randomly choose a stopping point so that the distribution of elements in the sampled portion is identical to choosing every element independently at random with probability 1/2. Then consider the maximum independent set I in the sample and count how many of these elements are in each value class, where value class i contains elements with values between 2^{i-1} and 2^i. Suppose that there are r_i elements of I in value class i. If a is the largest value class with at least one element, then we reserve the highest weight r_a seats for value class a, then the next r_{a-1} seats for value class $a-1$, and so on down to the smallest value class. Now when we see some element e from a set $A \in \mathcal{P}$, we add it to our solution if e is at least as valuable as the most valuable element from A in the sample, nothing else from A has been added to the solution, and there are still seats remaining among those reserved for the value class that contains $v(e)$. If we do choose to add e, then we assign it to the heaviest seat that has not been used yet among the seats reserved for value class i.

Theorem C.1 *The matroid reservation algorithm is* 16*-competitive for partition matroids, if there is no value class with exactly one element from the optimal solution in that class.*

Proof: Given a set $A \in \mathcal{P}$, let $m(A) = \text{argmax}_{e\in A}\, v(e)$ be the element in the set of maximum value. Let $P_i = \{A \in \mathcal{P} : v(m(A)) \in [2^{i-1}, 2^i)\}$, and let $u_i = |P_i|$. Note that u_i is the number of seats used for elements in value class i by the optimal solution. Let f_i be the number of seats into which the reservation algorithm places elements with value in value class i. For each set $A \in P_i$, we define two random variables: let X_A be 1 if the most valuable element from A appeared in the sample and 0 otherwise, and let Y_A be 1 if the second most valuable element from A appeared in the sample and 0 otherwise. Clearly $f_i \geq \min\{\sum_{A\in P_i} X_A, \sum_{A\in P_i} Y_A(1-X_A)\}$ since the algorithm reserves at least $\sum_{A\in P_i} X_A$ seats and tries to put at least $\sum_{A\in P_i} Y_A(1-X_A)$ elements from value class i into seats. Let $Y_i = \{Y_A : A \in P_i\}$ and let $X_i = \{X_A : A \in P_i\}$ be the set of Y and X variables. Taking expectations, we see that $\mathbf{E}[f_i] \geq \mathbf{E}\big[\min\{\sum_{A\in P_i} X_A, \sum_{A\in P_i} Y_A(1-X_A)\}\big] = \mathbf{E}_{Y_i}\big[\mathbf{E}_{X_i}\big[\min\{\sum_{A\in P_i} X_A, \sum_{A\in P_i} Y_A(1-X_A)\}|Y_i\big]\big]$.

When the Y values are fixed, we claim that $\mathbf{E}_{X_i}\big[\min\{\sum_{A\in P_i} X_A, \sum_{A\in P_i} Y_A(1-X_A)\}|Y_i\big] \geq \frac{1}{4}\sum_{A\in P_i} Y_A$. We consider three cases depending on $\sum_{u\in P_i} Y_A$. If this sum is 0 then the claim is trivially true. If the sum is at least 2 then there are at least two sets in P_i with Y value equal to 1, so

$$\begin{aligned}&\mathbf{E}_{X_i}\left[\min\{\sum_{A\in P_i} X_A, \sum_{A\in P_i} Y_A(1-X_A)\}|Y_i\right]\\ &\geq \mathbf{E}_{X_i}\left[\min\{\sum_{A\in P_i} Y_A X_A, \sum_{A\in P_i} Y_A(1-X_A)\}|Y_i\right]\\ &\geq \frac{1}{4}\sum_{A\in P_i} Y_A.\end{aligned}$$

If the sum equals 1 then there is only one set $A \in P_i$ that has $Y_A = 1$. Since every value class has either zero or at least two elements from the optimal solution, this means that there is some other $A' \in P_i$ with $Y_{A'} = 0$. If $X_A = 0$ and $X_{A'} = 1$ then $\min\{\sum_{A\in P_i} X_A, \sum_{A\in P_i} Y_A(1 - X_A)\} \geq 1 = \sum_{A\in P_i} Y_A$, which implies that $\mathbf{E}_{X_i}\left[\min\{\sum_{A\in P_i} X_A, \sum_{A\in P_i} Y_A(1-X_A)\}|Y_i\right] \geq \frac{1}{4}\sum_{A\in P_i} Y_A$.

With this claim established, we have that $\mathbf{E}[f_i] \geq \mathbf{E}_{Y_i}\left[\frac{1}{4}\sum_{A\in P_i} Y_A\right] = \frac{1}{8}u_i$. So in expectation the reservation algorithm reserves at least $\frac{1}{8}$ as many seats for value class i the optimal solution does.

Now we show that the seats reserved by the algorithm are at least as heavy as the seats used by the optimal solution. In the optimal solution each set in the partition reserves a seat for the value class containing its most valuable element, and in the algorithm each set reserves a seat for either the value class containing its most valuable element or a smaller value class. Thus $\sum_{j=i}^{a} r_j \leq \sum_{j=i}^{a} u_j$ for all $i \in [a]$. Let $b_i = \sum_{j=i}^{a} r_j$ and let $o_i = \sum_{j=i}^{a} u_j$.

Let s_O be the seating function of the optimal solution, and let s_R be the seating function of the solution returned by the reservation algorithm. Let $\mathsf{OPT}_i = \sum_{j=o_{i-1}+1}^{o_{i-1}+u_i} v(s_O^{-1}(j))w(j)$ be the contribution of value class i to the optimal solution, so $\mathsf{OPT} = \sum_i \mathsf{OPT}_i$. Similarly, let $R_i = \sum_{j=b_{i-1}+1}^{b_{i-1}+f_i} v(s_R^{-1}(j))w(j)$ be the contribution of class i to the solution given by the algorithm. Then

$$\begin{aligned}
\mathbf{E}[R_i] &\geq \mathbf{E}\left[2^{i-1}\sum_{j=b_{i-1}+1}^{b_{i-1}+f_i} w(j)\right] \\
&\geq \mathbf{E}\left[2^{i-1}\sum_{j=o_{i-1}+1}^{o_{i-1}+f_i} w(j)\right] \\
&\geq \frac{1}{8}2^{i-1}\sum_{j=o_{i-1}+1}^{o_{i-1}+u_i} w(j) \\
&\geq \frac{1}{16}2^{i}\sum_{j=o_{i-1}+1}^{o_{i-1}+u_i} w(j) \geq \frac{1}{16}\mathsf{OPT}_i.
\end{aligned}$$

Thus by linearity of expectation the expected value of the reservation algorithm is at least 1/16 of the optimal value. ∎

If there are value classes that only have one element in the optimal solution, then the reservation algorithm might not work because if that one element appears in the sample then it will not appear in the final solution, and if it appears after the sample then there might not be a spot reserved for it.We handle this case by essentially running the normal secretary algorithm in parallel: with probability $\frac{16}{3e+16}$ we run the reservation algorithm, and with probability $\frac{3e}{3e+16}$ we just run the basic secretary algorithm and assign the element we choose to the heaviest seat. Call this algorithm $\mathcal{A}$.

Theorem C.2 *$\mathcal{A}$ is $(16+3e)$-competitive.*

Proof: Let H be the set of value classes that have at least 2 elements in the optimal solution, and let T be the other value classes. Let $\mathsf{OPT}_H = \sum_{i\in H}\mathsf{OPT}_i$, and similarly let $\mathsf{OPT}_T = \sum_{i\in T}\mathsf{OPT}_i$. Clearly $\mathsf{OPT} = \mathsf{OPT}_H + \mathsf{OPT}_T$. Be Theorem C.1 the reservation algorithm returns a solution with expected value at least $\frac{1}{16}\mathsf{OPT}_H$. Let $v_{\max} = \max_{e\in U} v(e)$ be the largest value, and let $w_{\max}$ be the weight of the heaviest seat. Since the value classes are defined by powers of two, it is easy to see that $v_{\max}w_{\max} \geq \frac{1}{3}\mathsf{OPT}_T$, and thus the basic secretary algorithm gets at least $\frac{1}{3e}\mathsf{OPT}_T$ in expectation. We conclude that the expected value of the solution returned by $\mathcal{A}$ is at least

$$\frac{3e}{3e+16}\cdot\frac{1}{3e}\cdot\mathsf{OPT}_T + \frac{16}{3e+16}\cdot\frac{1}{16}\cdot\mathsf{OPT}_H = \frac{1}{3e+16}\mathsf{OPT}$$

which completes the proof. ∎

The matroid reductions of Section 5 now immediately imply results for graphical, uniform, and low-and high-degree transversal matroids.

Corollary C.3 *There are algorithms for the weighted secretary problem on graphical, uniform, and low- and high-degree transversal matroids that have constant competitive ratio.*

D Discounted Matroid Secretary: General Matroids

Proof of Theorem 5.5. The algorithm and proof are similar to those of Theorem 4.4. For $c \geq 1$ let I_c be the interval $(2^{-i}d_{\max}, 2^{-(i-1)}d_{\max}]$ and let $P_c = \{i \in [n] : d(\pi(i)) \in I_c\}$. The algorithm works by first choosing a random discount class $c \in [\log(nk)]$ uniformly at random and then running the Threshold Price Algorithm of [BIK07, Section 3] only on P_c. As before, we first show that we can ignore discounts that are too small. Given a permutation $\pi \in \mathfrak{S}_n$, let $I_\pi = \operatorname{argmax}_{I\in\mathcal{I}} \sum_{i\in I} d(\pi(i))v(i)$ be the optimal independent

set, and let $I_\pi^c = \{i \in I_\pi : \pi(i) \in P_c\}$. Clearly

$$\begin{aligned}\mathbf{E}[\mathsf{OPT}] &= \frac{1}{n!}\left(\sum_{\pi\in\mathfrak{S}_n}\sum_{c=1}^{\infty}\sum_{i\in I_\pi^c} d(\pi(i))v(i)\right)\\ &= \frac{1}{n!}\left(\sum_{\pi\in\mathfrak{S}_n}\sum_{c=1}^{\log(nk)}\sum_{i\in I_\pi^c} d(\pi(i))v(i)\right.\\ &\qquad \left.+\sum_{\pi\in\mathfrak{S}_n}\sum_{c=\log(nk)+1}^{\infty}\sum_{i\in I_\pi^c} d(\pi(i))v(i)\right)\end{aligned}$$

Note that $\sum_{\pi\in\mathfrak{S}_n}\sum_{c=1}^{\log(nk)}\sum_{i\in I_\pi^c} d(\pi(i))v(i) \geq (n-1)!d_{\max}v_{\max}$, since in at least $(n-1)!$ permutations the most valuable element is put in the location with the largest discount. Also,

$$\begin{aligned}&\sum_{\pi\in\mathfrak{S}_n}\sum_{c=\log(nk)+1}^{\infty}\sum_{i\in I_\pi^c} d(\pi(i))v(i)\\ &\qquad \leq n!kv_{\max}d_{\max}/(nk) \leq (n-1)!d_{\max}v_{\max}, \quad \text{(D.5)}\end{aligned}$$

since at most k elements can be picked and they are all in a location with discount at most $d_{\max}/(nk)$. So if we only consider the large discount classes, we have that

$$\begin{aligned}\mathbf{E}[\mathsf{OPT}]/2 &\leq \frac{1}{n!}\sum_{\pi\in\mathfrak{S}_n}\sum_{c=1}^{\log(nk)}\sum_{i\in I_\pi^c} d(\pi(i))v(i)\\ &\leq \frac{1}{n!}\sum_{\pi\in\mathfrak{S}_n}\sum_{c=1}^{\log(nk)}\max_{I\in\mathcal{I}}\left(\sum_{i\in I:\pi(i)\in P_c} d(\pi(i))v(i)\right)\\ &\leq \sum_{c=1}^{\log(nk)} 2^{-(c-1)}\frac{1}{n!}\sum_{\pi\in\mathfrak{S}_n}\max_{I\in\mathcal{I}}\sum_{i\in I:\pi(i)\in P_c} v(i)\\ &\leq \sum_{c=1}^{\log(nk)} 2^{-(c-1)}\mathbf{E}\left[\max_{I\in\mathcal{I}}\sum_{i\in I:\pi(i)\in P_c} v(i)\right]\end{aligned}$$

We know from [BIK07, Theorem 3.2] that the Threshold Prince Algorithm is $32\lceil\log k\rceil$-competitive. It is easy to see (by conditioning on the set of elements which actually appear in the locations in P_c) that the threshold price algorithm is actually $32\lceil\log k\rceil$ competitive when restricted to locations in P_c, i.e. in expectation it returns an independent set I the sum of whose values is at least $\frac{1}{32\lceil\log k\rceil}\mathbf{E}\left[\max_{I\in\mathcal{I}}\sum_{i\in I:\pi(i)\in P_c} v(i)\right]$. This means that the expected value of the algorithm is at least

$$\begin{aligned}&\sum_{c=1}^{\log(nk)}\frac{1}{\log(nk)}2^{-c}\frac{1}{32\lceil\log k\rceil}\mathbf{E}\left[\max_{I\in\mathcal{I}}\sum_{i\in I:\pi(i)\in P_c} v(i)\right]\\ &\qquad\geq \frac{1}{128\log(nk)\lceil\log k\rceil}\mathbf{E}[\mathsf{OPT}]\end{aligned}$$

Since $k \leq n$ this implies that the algorithm is $O(\log n \log k)$-competitive as claimed. □