A Knapsack Secretary Problem with Applications

Moshe Babaioff[1], Nicole Immorlica[2], David Kempe[3], and Robert Kleinberg[4]

[1] UC Berkeley School of Information. `moshe@ischool.berkeley.edu`. Supported by NSF ITR Award ANI-0331659.
[2] Microsoft Research. `nickle@microsoft.com`
[3] University of Southern California, Department of Computer Science. `dkempe@usc.edu`. Work supported in part by NSF CAREER Award 0545855.
[4] Cornell University, Department of Computer Science. `rdk@cs.cornell.edu`. Partially supported by an NSF Mathematical Sciences Postdoctoral Research Fellowship. Portions of this work were completed while the author was a postdoctoral fellow at UC Berkeley.

Abstract. We consider situations in which a decision-maker with a fixed budget faces a sequence of options, each with a cost and a value, and must select a subset of them online so as to maximize the total value. Such situations arise in many contexts, e.g., hiring workers, scheduling jobs, and bidding in sponsored search auctions.
This problem, often called the *online knapsack problem*, is known to be inapproximable. Therefore, we make the enabling assumption that elements arrive in a *random* order. Hence our problem can be thought of as a weighted version of the classical *secretary problem*, which we call the *knapsack secretary problem*. Using the random-order assumption, we design a constant-competitive algorithm for arbitrary weights and values, as well as a e-competitive algorithm for the special case when all weights are equal (i.e., the *multiple-choice secretary problem*). In contrast to previous work on online knapsack problems, we do not assume any knowledge regarding the distribution of weights and values beyond the fact that the order is random.

1 Introduction

Allocation of resources under uncertainty is a very common problem in many real-world scenarios. Employers have to decide whether or not to hire candidates, not knowing whether future candidates will be stronger or more desirable. Machines need to decide whether to accept jobs without knowledge of the importance or profitability of future jobs. Consulting companies must decide which jobs to take on, not knowing the revenue and resources associated with potential future requests.

More recently, online auctions have proved to be a very important resource allocation problem. Advertising auctions in particular provide the main source of monetization for a variety of Internet services including search engines, blogs, and social networking sites. Additionally, they are the main source of customer acquisition for a wide array of small online businesses, the so-called "mom and pop shops" of the networked world. In bidding for the right to appear on a web page (such as a search engine), advertisers have to trade off between large numbers of parameters, including keywords and viewer attributes. In this scenario, an advertiser may be able to estimate accurately the bid required to win a particular auction, and the benefit either in direct revenue or name recognition to be gained, but may not know about the trade off for future auctions.

All of these problems involve an online scenario, wherein an algorithm has to make decisions on whether to accept an offer (such as a candidate, job, or a bidding opportunity), based solely on the required resource investment (or *weight*) w and projected *value* v of the current offer, without knowledge of the weights or values of future offers. The total weight of all selected offers may not exceed a given budget W. Thus, the problem we are concerned with is an online knapsack problem. In general, this problem does not permit any good competitive ratio, as evidenced by trivial bad examples. Instead, we focus on the case where the offers arrive in a uniformly *random* order.

Summary of Results: In this model, we prove two results: for the case of general weights and values, we give a constant-competitive online algorithm (specifically, it is $10e$-competitive). For the special case where all the weights are uniform, and the weight constraint thus poses a constraint on the total *number* of offers that can be accepted, we improve the approximation factor to e, via two simple and natural algorithms.

Secretary Problems: When the weights are uniform and equal to the weight constraint, our problem reduces to the famous *secretary problem*, or the problem of selecting online an element of maximum value in a randomly-ordered sequence. This problem was first introduced by Dynkin [9] in 1963. His paper gives an algorithm which selects the maximum value element with probability that tends to $1/e$ as n tends to infinity and hence is e-competitive. Many generalizations of this problem have been studied in the literature. In one natural generalization, Kleinberg [11] considers the multiple-choice secretary problem in which k elements need to be selected and the goal is to maximize the combined value (sum) of the selected elements. Kleinberg presents an asymptotically optimal $1/(1-5/\sqrt{k})$-competitive algorithm for this problem. Another closely related generalization considered in the literature is the *matroid secretary problem*, introduced by Babaioff et al. [2], in which the elements of a weighted matroid arrive in a random order. As each element is observed, the algorithm makes an irrevocable decision to choose it or skip it, with the constraint that the chosen elements must constitute an independent set. Again, the objective is to maximize the combined weight of the chosen elements. Babaioff et al. give an $O(\log k)$-competitive algorithm for the matroid secretary problem, where k is the rank of the matroid, as well as constant-competitive algorithms for several specific matroids.

In this paper, we study both the multiple-choice secretary problem and a weighted generalization, which we call the *knapsack secretary problem*. The multiple-choice secretary problem is a special case of the matroid secretary problem (for the truncated uniform matroid). We show how to apply an intuitive algorithmic idea proposed by Babaioff et al. [2] to get a e-competitive algorithm for this problem for any k. Hence, our result improves upon the competitive ratio of the algorithm by Kleinberg [11] for small k and is significantly simpler. The knapsack secretary problem, on the other hand, can not be interpreted as a matroid secretary problem, and hence none of the previous results apply. In this paper, we give the first constant-competitive algorithm for this problem, using intuition from the standard 2-approximation algorithm for the offline knapsack problem.

Knapsack Problems: Our work builds upon the literature for knapsack problems. It is well known that the NP-complete (offline) knapsack problem admits an FPTAS as well as a simple 2-approximation, whereas the online knapsack problem is inapproximable to within any non-trivial multiplicative factor. Assuming that the density (value to weight ratio) of every element is in a known range $[L, U]$, and that each weight is much smaller than the capacity of the knapsack (or that the packing is allowed to be fractional), Buchbinder and Naor [4, 5] give an algorithm with a multiplicative competitive ratio of $O(\log(U/L))$ for *online knapsack* based on a general online primal-dual framework. They also show an $\Omega(\log(U/L))$ lower bound on the competitive ratio of any algorithm under such assumptions.

Several papers have also considered a *stochastic online knapsack* problem [12, 13] in which the value and/or weight of elements are drawn according to a known distribution. These papers provide algorithms with an additive approximation ratio of $\Theta(\log n)$ as well as showing that no online algorithm can achieve a constant additive approximation. Dean et al. [7, 8] consider a *stochastic offline knapsack* problem where the algorithm knows the values and the distribution of the weights of the elements.

They present an involved way for choosing the order of the elements so as to achieve a constant-competitive outcome in the multiplicative sense. The main difficulty in their model is that the weight of an element is not revealed until it is actually selected.

Our results show that a constant-competitive algorithm exists for any sequence *when elements arrive in a random order*. The random order assumption allows us to eliminate all assumptions from previous papers, e.g., that elements have small weights [4, 5], and densities are bounded [4, 5] or drawn according to a known distribution [7, 8, 12, 13].[5] In return, we are able to design a constant-competitive online algorithm for our setting. In contrast, for the online setting of Buchbinder and Naor, there is a super-constant lower bound of $\Omega(\ln(U/L))$ for a worst-case order of arrivals [4, 5].

Sponsored Search: Several recent papers have considered applications of the knapsack problem to auction design. Aggarwal and Hartline [1] design truthful auctions which are revenue competitive when the auctioneer is constrained to choose agents with private values and publicly known weights that fit into a knapsack. Knapsack algorithms have also been used to design bidding strategies for budget-constrained advertisers in sponsored search auctions. That the bidding problem in such settings is similar to knapsack was first noted by Borgs et al. [3] (who considered using knapsack to model slot selection) and Rusmevichientong and Williamson [16] (who considered using stochastic knapsack to model keyword selection). The bidding problem was further studied in papers by Feldman et al. [10] and Muthukrishnan et al. [15] which consider the problem of slot selection in more complicated settings, including interactions between keywords and stochastic information. All these papers assume that the set of keywords and distributional information regarding values and weights are known upfront by the algorithm; hence the algorithms they develop are inspired by offline knapsack problems. Recently, Chakrabarty et al. [6] modeled the bidding problem using online knapsack. Under the same assumptions as the paper of Buchbinder and Naor [4, 5] mentioned above, Chakrabarty et al. design a $(\ln(U/L) + 1)$-competitive online algorithm for a worst case sequence of keywords.

Outline of paper: In Section 2, we introduce a formal model for the knapsack secretary problem. We then give a pair of e-competitive algorithms for the unweighted knapsack secretary problem in Section 3.

[5] In contrast to the Dean et al. [7, 8] models, our model and the others mentioned make the stronger assumption that the weights of elements are learned *before* deciding whether or not to select them.

Finally, in Section 4, we design a constant-competitive algorithm for the general case.

2 Model

In formalizing the resource allocation problem, we will adopt the terminology of the *secretary problem*, and think of our problem as a *weighted secretary problem.* A set $U = \{1, \ldots, n\}$ of n elements or *secretaries* each have non-negative *weight* $w(i)$ and *value* $v(i)$. We extend the notation to sets by writing $w(S) := \sum_{i \in S} w(i)$ and $v(S) := \sum_{i \in S} v(i)$.

The algorithm will be given a *weight bound* W, and must select, in an online fashion, a set $S \subseteq U$ of secretaries (approximately) solving the following knapsack problem:

$$\text{Maximize} \sum_{i \in S} v(i) \qquad \text{subject to} \qquad \sum_{i \in S} w(i) \leq W. \tag{1}$$

We assume that the secretaries in U are presented to the algorithm in a uniformly random order. In order to be able to number the elements by their arrival order, we assume that the actual weights and values are obtained as $v = v_0 \circ \pi, w = w_0 \circ \pi$, where π is a uniformly random permutation of n elements, and v_0, w_0 are arbitrary initial weight and value functions. For simplicity, we also assume that no two secretaries have the same values $v(i), v(j)$. This is easy to ensure, by fixing a random (but consistent) tie-breaking between elements of the same value, based for instance on the identifier of the element.[6]

The algorithm is online in the following sense: initially, the algorithm knows only n, the total number of secretaries, but knows nothing about the distribution of weights or values. Whenever a secretary i arrives, the algorithm learns its weight $w(i)$ and value $v(i)$. It must then irrevocably decide whether to *select* i or *pass*: a selected secretary cannot later be discarded, nor can a passed secretary be added. Thus, the algorithm maintains a set S of currently selected secretaries, which grows over the course of the execution, but must always satisfy $w(S) \leq W$.

Clearly, this setting does not permit the design of an optimal algorithm. Hence, we look for algorithms which are constant-competitive in that the expected value of the selected set S is within a constant of the optimum value. More precisely, we say an algorithm is α-competitive for the weighted secretary problem if for any initial weight and value functions

[6] Note that such a tie-breaking can be accomplished in polynomial time.

v_0, w_0

$$\alpha \cdot \mathrm{E}\left[v(S)\right] \geq v(S^*),$$

where S^* is the optimal solution to Program 1 and the expectation is over all permutations π.

Note that this is a generalization of the classical secretary problem of Dynkin [9]. In the classical secretary problem, all weights are one (i.e., $w(i) = 1$ for all i) and the weight bound W is also one; thus, the algorithm is to select exactly one secretary. Dynkin gives a e-competitive algorithm for this special case. Our formulation can also be used to capture the k-secretary problem by setting all weights equal to one and the weight bound W equal to k. This case has been studied by Kleinberg [11], who gave a $1/(1 - 5/\sqrt{k})$-competitive algorithm.

In the following sections, we first present two algorithms for the k-secretary problem. Our algorithms are simpler than those of Kleinberg and show that there is a e-competitive algorithm for *all* k (Kleinberg's result is strictly worse than e for small k). We then present a constant-competitive algorithm for the general case of weighted secretaries, although the constant is worse than that of k-secretaries.

3 The Unweighted Case

In this section we present two simple algorithms for the unweighted case (i.e., the multiple-choice secretary problem), in which all weights $w(i)$ are equal to 1 and the knapsack capacity W is equal to k. Both algorithms will achieve a competitive guarantee no worse than e. While the second algorithm, called the "optimistic algorithm" is perhaps more natural (and our analysis is almost certainly not tight), the first algorithm, called the "virtual algorithm", has a significantly simpler analysis, yielding essentially a tight bound on its performance.

Both algorithms are based on the same idea of a sampling period of $t \in \{k + 1, \ldots, n\}$ steps (during which the algorithm passes on all candidates), followed by hiring some of the secretaries for the remaining $n - t$ steps. We call t the *threshold time* of the algorithms, and denote the set of sampled elements by T. We leave t unspecified for now; after analyzing the algorithm, we will specify the optimal value of t, which will be approximately n/e.

Both algorithms use the first t time steps to assemble a *reference set* R, consisting of the k elements with the largest $v(i)$ values seen during the first t steps. These elements are kept for comparison, but *not selected.* Subsequently, when an element $i > t$ with value $v(i)$ is observed, a decision of whether to select i into the set S is made based on $v(i)$ and R, and the set R is possibly updated. At any given time, let $j_1, j_2, \ldots, j_{|R|}$ be the elements of R, sorted by decreasing $v(j_i)$.

Virtual: In the virtual algorithm, i is selected if and only if $v(i) > v(j_k)$, and $j_k \leq t$ (j_k is in the sample). In addition, whenever $v(i) > v(j_k)$ (regardless of whether $j_k \leq t$), element i is added to R, while element j_k is removed from R.

Thus, R will always contain the best k elements seen so far (in particular, $|R| = k$), and i is selected if and only if its value exceeds that of the k^{th} best element seen so far, and the k^{th} best element was seen during the sampling period.

Optimistic: In the optimistic algorithm, i is selected if and only if $v(i) > v(j_{|R|})$. Whenever i is selected, $j_{|R|}$ is removed from the set R, but no new elements are ever added to R. Thus, intuitively, elements are selected when they beat one of the remaining reference points from R.

We call this algorithm "optimistic" because it removes the reference point $j_{|R|}$ even if $v(i)$ exceeds, say, $v(j_1)$. Thus, it implicitly assumes that it will see additional very valuable elements in the future, which will be added when their values exceed those of the remaining, more valuable, j_i.

We first observe that neither algorithm ever selects more than k secretaries. Each selection involves the removal of a sample $j_i \in R \cap T$ from R, and no elements from T are ever added to R by either algorithm after time t. Since R starts with only k samples, no more than k elements can be selected.

Next, we prove that both the virtual and the optimistic Algorithm are e-competitive, if $t = \lfloor n/e \rfloor$ elements are sampled.

Theorem 1. *The competitive ratio of both the Virtual and the Optimistic Algorithm approaches* e *as* n *tends to infinity, when the algorithms sample* $t = \lfloor n/e \rfloor$ *elements.*

The proof of the theorem for both algorithms follows from stronger lemmas, establishing that *each* of the top k elements is selected with probability at least $1/e$. Specifically, let $v_1^*, v_2^*, \ldots, v_k^*$ denote the k largest elements of the set $\{v(1), v(2), \ldots, v(n)\}$, and for $a = 1, 2, \ldots, k$ let $i_a^* = v^{-1}(v_a^*)$ be the index in the sequence $v(i)$ at which v_a^* appeared. We will then establish the following lemmas:

Lemma 1. *For all $a \le k$, the probability that the virtual algorithm selects element v_a^* is*

$$\mathrm{Prob}[i_a^* \in S] \ge \tfrac{t}{n} \ln(n/t).$$

Lemma 2. *For all $a \le k$, the probability that the optimistic algorithm selects element v_a^* is*

$$\mathrm{Prob}[i_a^* \in S] \ge \tfrac{t}{n} \ln(n/t).$$

Proof of Theorem 1. The theorem follows immediately from these two lemmas, as the expected gain of the algorithm is

$$\mathrm{E}\,[v(S)] \ge \textstyle\sum_{a=1}^{k} \mathrm{Prob}[i_a^* \in S] \cdot v_a^* > \tfrac{t}{n} \ln(n/t) \cdot v(S^*).$$

$\frac{t}{n} \ln(n/t)$ is maximized for $t = n/e$, and setting $t = \lfloor n/e \rfloor$ gives us that $t/n \to 1/e$ as $n \to \infty$. Thus, the algorithms' competitive ratios approach e as n tends to infinity. ■

The proof of Lemma 1 turns out to be surprisingly simple and elegant, while the proof of Lemma 2 for the optimistic algorithm is significantly more complex, and will be given in the full version of this paper.

Proof of Lemma 1. If v_a^* is observed at time $i_a^* = i > t$, it will be selected if and only if the k^{th} smallest element of R at that time was sampled at or before time t. Because the permutation is uniformly random, this happens with probability $t/(i-1)$. Each i is equally likely to be the time at which v_a^* is observed, so the probability of selecting v_a^* is

$$\mathrm{Prob}[i_a^* \in S] = \textstyle\sum_{i=t+1}^{n} \frac{1}{n} \cdot \frac{t}{i-1} = \frac{t}{n} \sum_{i=t+1}^{n} \frac{1}{i-1} > \frac{t}{n} \int_t^n \frac{dx}{x} = \frac{t}{n} \ln\left(\frac{n}{t}\right).$$

■

Notice that the proof of Lemma 1 is essentially tight. Each of the top k elements is selected with probability approaching $1/e$ in the limit for our choice of t.

4 The Weighted Case

In this section, we present an algorithm for the weighted case, with a competitive ratio of $10e$. The algorithm is based on the familiar paradigm of sampling a constant fraction of the input and using the sample to define a selection criterion which is then applied to the subsequent elements observed by the algorithm. One complication which arises in designing algorithms for the weighted case is the need to address at least two cases: either there is a single element (or, more generally, a bounded number of elements) whose value constitutes a constant fraction of the optimal knapsack solution, or there is no such element[7]. In the former case(s), we use a selection criterion based on the values of elements but ignoring their sizes. In the latter case, we use a selection criterion based on the *value density*, i.e., the ratio of value to weight. To incorporate both cases, we randomize the selection criterion.

4.1 Notation

For $i \in U$, we define the *value density* (or simply "density") of i to be the ratio

$$\rho(i) = \frac{v(i)}{w(i)}.$$

We will assume throughout this section that distinct elements of U have distinct densities; this assumption is justified for the same reason our assumption of distinct values is justified. (See Section 2.) If $Q \subseteq U$ and $x > 0$, it will be useful to define the "optimum fractional packing of elements of Q into a knapsack of size x." This is defined to be a vector of weights $(y_Q^{(x)}(i))_{i=1}^n$ which is a solution of the following linear program (that is, $y_Q^{(x)}(i) = y(i)$).

$$\begin{array}{lll} \max & \sum_{i=1}^n v(i)y(i) & \\ \text{s.t.} & \sum_{i=1}^n w(i)y(i) \le x & \\ & y(i) = 0 & \forall i \notin Q \\ & y(i) \in [0,1] & \forall i. \end{array} \tag{2}$$

The reader may verify the following easy fact about $y_Q^{(x)}(i)$: there exists a *threshold density* $\rho_Q^{(x)}$ such that $y_Q^{(x)}(i) = 1$ for all $i \in Q$ such that $\rho(i) > \rho_Q^{(x)}$ and $y_Q^{(x)}(i) = 0$ for all $i \in Q$ such that $\rho(i) < \rho_Q^{(x)}$. Finally, for a set $R \subseteq U$ we will define $v_Q^{(x)}(R), w_Q^{(x)}(R)$ by

$$v_Q^{(x)}(R) = \sum_{i \in R} v(i) y_Q^{(x)}(i)$$
$$w_Q^{(x)}(R) = \sum_{i \in R} w(i) y_Q^{(x)}(i).$$

[7] This type of case analysis is reminiscent of the case analysis which underlies the design of polynomial-time approximation schemes for the offline version of the knapsack problem.

4.2 The algorithm

For convenience, we assume in this section that $W = 1$. (To reduce from the general case to the $W = 1$ case, simply rescale the weight of each element by a factor of $1/W$.) Our algorithm begins by sampling a random number $a \in \{0, 1, 2, 3, 4\}$ from the uniform distribution. The case $a = 4$ is a special case which will be treated in the following paragraph. If $0 \leq a \leq 3$, then the algorithm sets $k = 3^a$ and runs the k-secretary algorithm from Section 3 (with $t = \lfloor n/e \rfloor$) to select at most k elements. If the k-secretary algorithm selects an element i whose weight $w(i)$ is greater than $1/k$, we override this decision and do not select the element.

If $a = 4$, our algorithm operates as follows. It samples a random $t \in \{1, 2, \ldots, n\}$ from the binomial distribution $B(n, 1/2)$, i.e. the distribution of the number of heads observed when a fair coin is tossed n times. Let $X = \{1, 2, \ldots, t\}$ and $Y = \{t + 1, t + 2, \ldots, n\}$. For every element $i \in X$, the algorithm observes $v(i)$ and $w(i)$ but does not select i. It then sets $\hat{\rho} = \rho_X^{(1/2)}$ and selects every element $i \in Y$ which satisfies $w(i) \leq 3^{-4}$, $\rho(i) \geq \hat{\rho}$, and $w(S_{<i} \cup \{i\}) \leq 1$, where $S_{<i}$ denotes the set of elements which were already selected by the algorithm before observing i.

4.3 Analysis of the algorithm

Theorem 2. *The algorithm in Section 4.2 is* $(10e)$*-competitive.*

Proof. Let $\mathsf{OPT} \subseteq U$ denote the maximum-value knapsack solution, and suppose that $i_1, i_2, \ldots, i_m$ are the elements of OPT arranged in decreasing order of weight. Partition OPT into five sets $B_0, B_1, \ldots, B_4$. For $0 \leq j \leq 3$,

$$B_j = \{i_\ell \,|\, 3^j \leq \ell < 3^{j+1}\},$$

while for $j = 4$, $B_4 = \{i_{81}, i_{82}, \ldots, i_m\}$. Let $b_j = v(B_j)$ for $0 \leq j \leq 4$.

Let S denote the set of elements selected by the algorithm. For $0 \leq j \leq 4$, define

$$g_j = E\left[v(S) \mid a = j\right]$$

where a denotes the random element of $\{0, 1, 2, 3, 4\}$ sampled in the first step of the algorithm. In Lemmas 3 and 4 below, we prove that $b_j \leq 2eg_j$

for $0 \le j \le 4$. Summing over j, we obtain:

$$\begin{aligned} v(\mathsf{OPT}) &= b_0 + b_1 + b_2 + b_3 + b_4 \\ &\le 2e(g_0 + g_1 + g_2 + g_3 + g_4) \\ &= (10e) \sum_{j=0}^{4} \mathrm{Prob}[a = j] g_j \\ &= 10e\, E[v(S)]. \end{aligned}$$

This establishes the theorem. □

Lemma 3. *For $0 \le j \le 3$, $b_j \le 2eg_j$.*

Proof. Let $k = 3^j$. Recall that every element $i \in B_j$ appears in at least the k^{th} position on a list of elements of OPT arranged in decreasing order of weight. Since the sum of the weights of all elements of OPT is at most 1, we have that $w(i) \le 1/k$ for every $i \in B_j$. Let $Q = \{i \in U \mid w(i) \le 1/k\}$, and let R be the maximum-value k-element subset of Q. Since $B_j \subseteq Q$ and $|B_j| \le 2k$, we have $v(B_j) \le 2v(R)$. On the other hand, Theorem 1 implies that $g_j \ge v(R)/e$. The lemma follows by combining these two bounds. □

Lemma 4. *$b_4 \le 2eg_4$.*

Proof. Assuming the algorithm chooses $a = 4$, recall that it splits the input into a "sample set" $X = \{1, 2, \ldots, t\}$ and its complement $Y = \{t+1, \ldots, n\}$, where t is a random sample from the binomial distribution $B(n, 1/2)$. Recall that in the case $a = 4$, the algorithm aims to fill the knapsack with multiple items of weight at most 1/81, and value density at least equal to the value density of the optimal solution for the sample (and a knapsack of size 1/2). Thus, let $Q \subseteq U$ consist of all elements $i \in U$ such that $w_0(i) \le 1/81$. We will show that with sufficiently high constant probability, the algorithm obtains a "representative" sample, in the sense that the optimal value density estimated from X is bounded from above and below in terms of the optimal value density for all of Q (with different knapsack sizes). This in turn will imply that each element of Q is picked by the algorithm with constant probability, more specifically, with probability at least 0.3.

To obtain sufficiently high probability, we rely on the independence of membership in X between elements, which in turn allows us to apply Chernoff Bounds. Recall that we encoded the random ordering of the input by assuming that there exists a fixed pair of functions v_0, w_0 and a uniformly random permutation π on U, such that $v = v_0 \circ \pi$, $w = w_0 \circ \pi$. This implies that, conditional on the value of t, $\pi^{-1}(X)$ is a uniformly-random t-element subset of U. Since t itself has the same distribution as the cardinality of a uniformly-random subset of U, it follows that $\pi^{-1}(X)$ is a uniformly-random subset of U. For each $i \in U$, if we define

$$\zeta_i = \begin{cases} 1 \text{ if } \pi(i) \in X \\ 0 \text{ otherwise,} \end{cases}$$

then the random variables ζ_i are mutually independent, each uniformly distributed in $\{0, 1\}$.

Since $B_4 \subseteq Q$ and $w(B_4) \leq 1$,

$$b_4 \leq v_Q^{(1)}(Q) \leq \frac{4}{3} v_Q^{(3/4)}(Q). \tag{3}$$

For every j such that $y_{\pi(Q)}^{(3/4)}(\pi(j)) > 0$ we will prove that $\text{Prob}[\pi(j) \in S \,|\, a = 4] > 0.3$. This implies the first inequality in the following line, whose remaining steps are clear from the definitions.

$$v_{\pi(Q)}^{(3/4)}(\pi(Q)) < E\left[\tfrac{10}{3} v_{\pi(Q)}^{(3/4)}(S) \,|\, a = 4\right] \leq \tfrac{10}{3} E[v(S) \,|\, a = 4] = \tfrac{10}{3} g_4. \tag{4}$$

Combining (3) and (4) we will have derived $b_4 \leq (40/9) g_4 < 2e g_4$, thus establishing the lemma.

Note that for all $i \in U$, $x > 0$, the number $y_{\pi(Q)}^{(x)}(\pi(i))$ does not depend on the random permutation π, since it is the i-th component of the solution of linear program (2) with v_0 and w_0 in place of v and w, and the solution to the linear program does not depend on π. We will use the notation $y(i, x)$ as shorthand for $y_{\pi(Q)}^{(x)}(\pi(i))$. Fix any $j \in Q$. We will show that j will be picked by the algorithm with probability at least 0.3. To prove this, we will upper and lower bound the total weight of $\pi(Q)$ (scaled by the fractional solutions for knapsacks of different sizes) seen in X and Y. This will allow us to reason that j will have density exceeding $\hat{\rho}$, and there will still be room in S by the time j is encountered.

We will reason about the expected fractional weight of items other than j in X in a knapsack of size 3/4, and of items other than j in Y in a knapsack of size 3/2. Formally, we define the random variables

$$Z_1 = w_{\pi(Q)}^{(3/4)}(X \setminus \{\pi(j)\}) = \sum_{i \in Q \setminus \{j\}} w_0(i) y(i, 3/4) \zeta_i \tag{5}$$

$$Z_2 = w_{\pi(Q)}^{(3/2)}(Y \setminus \{\pi(j)\}) = \sum_{i \in Q \setminus \{j\}} w_0(i) y(i, 3/2)(1 - \zeta_i) \tag{6}$$

Since Z_1, Z_2 are sums of independent random variables taking values in the interval $[0, 1/81]$, we can use the following form of the Chernoff bound, obtained from standard forms [14] by simple scaling: If $z_1, z_2, \ldots, z_n$ are independent random variables taking values in an interval $[0, z_{\max}]$ and if $Z = \sum_{i=1}^{n} z_i$, $\mu = E[Z]$, then for all $\delta > 0$,

$$\text{Prob}[Z \geq (1+\delta)\mu] < \exp\left(-\frac{\mu}{z_{\max}}\left[(1+\delta)\ln(1+\delta) - \delta\right]\right).$$

Because the expectations of Z_1 and Z_2 are

$$E[Z_1] = \tfrac{1}{2} w^{(3/4)}_{\pi(Q)}(\pi(Q) \setminus \{\pi(j)\}) = \tfrac{1}{2}\left(\tfrac{3}{4} - w_0(j) y(j, 3/4)\right) \in \left[\tfrac{3}{8} - \tfrac{1}{162}, \tfrac{3}{8}\right],$$
$$E[Z_2] = \tfrac{1}{2} w^{(3/2)}_{\pi(Q)}(\pi(Q) \setminus \{\pi(j)\}) = \tfrac{1}{2}\left(\tfrac{3}{2} - w_0(j) y(j, 3/2)\right) \in \left[\tfrac{3}{4} - \tfrac{1}{162}, \tfrac{3}{4}\right],$$

applying the Chernoff Bound to Z_1 and Z_2 with $z_{\max} = 1/81, \delta = \frac{1}{3} - \frac{8}{243}$ yields $\text{Prob}[Z_1 \geq 1/2 - 1/81] < 0.3$ and $\text{Prob}[Z_2 \geq 1 - 2/81] < 0.1$.

Let $\mathcal{E}$ denote the event that $Z_1 < \frac{1}{2} - \frac{1}{81}$ and $Z_2 < 1 - \frac{2}{81}$. By a union bound, $\text{Prob}[\mathcal{E} \mid a = 4] > 0.6$. Conditional on the event $\mathcal{E}$ (and on the event that $a = 4$), the element $\pi(j)$ can add no more than $1/81$ to the weight of X or Y (whichever one it belongs to). Hence, $w^{(3/4)}_{\pi(Q)}(X) < 1/2$ and $w^{(3/2)}_{\pi(Q)}(Y) < 1 - \frac{1}{81}$, which in turn implies $w^{(3/2)}_{\pi(Q)}(X) > 1/2 > w^{(3/4)}_{\pi(Q)}(X)$, since every element of $\pi(Q)$ belongs to either X or Y and $w^{(3/2)}_{\pi(Q)}(\pi(Q)) = 3/2$. Because the threshold density for a fractionally packed knapsack with larger capacity cannot be larger than for a knapsack with smaller capacity, the above bounds on the weight imply that

$$\rho^{(3/4)}_{\pi(Q)} \geq \rho^{(1/2)}_{X} \geq \rho^{(3/2)}_{\pi(Q)}. \tag{7}$$

Let S^+ denote the set of all elements of $Y \setminus \{\pi(j)\}$ whose value density is greater than or equal to $\hat{\rho} = \rho^{(1/2)}_X$. (Note that the algorithm will pick every element of S^+ that it sees until it runs out of capacity, and it will not pick any element which does not belong to S^+ except possibly $\pi(j)$.) We claim that the combined size of the elements of S^+ is at most $1 - \frac{1}{81}$. This can be seen from the fact that for all but at most one $i \in S^+$, the coefficient $y^{(3/2)}_{\pi(Q)}(i)$ is equal to 1. Hence the combined size of all the elements of S^+ is bounded above by

$$\tfrac{1}{81} + w^{(3/2)}_{\pi(Q)}(Y \setminus \{\pi(j)\}) = \tfrac{1}{81} + Z_2 < 1 - \tfrac{1}{81},$$

from which it follows that the algorithm does not run out of room in its knapsack before encountering $\pi(j)$. If $y(j, 3/4) > 0$, then $\rho(\pi(j)) \geq \rho^{(3/4)}_{\pi(Q)}$

and (7) implies that $\rho(\pi(j)) \geq \hat{\rho}$. Thus, the algorithm will select $\pi(j)$ if $\pi(j) \in Y$. Finally, note that the event $\pi(j) \in Y$ is independent of $\mathcal{E}$, so

$$\text{Prob}[\pi(j) \in S \mid \mathcal{E} \wedge (a = 4)] = \text{Prob}[\pi(j) \in Y \mid \mathcal{E} \wedge (a = 4)] = \tfrac{1}{2}.$$

Combining this with the bound $\text{Prob}[\mathcal{E} \mid a = 4] > 0.6$ established earlier, we obtain

$$\text{Prob}[\pi(j) \in S \mid a = 4] > 0.3,$$

which completes the proof of the lemma. □

5 Conclusions

In this paper, we have presented algorithms for a knapsack version of the secretary problem, in which an algorithm has to select, in an online fashion, a maximum-value subset from among the randomly ordered items of a knapsack problem. We gave a constant-competitive algorithm in this model, as well as a e-approximation for the k-secretary problem, in which all items have identical weights.

The competitive ratios we obtain are certainly not tight, and it appears that the analysis for the "optimistic algorithm" is not tight, either. Determining the exact competitive ratio for this algorithm, as well as improving the algorithm for the knapsack problem, are appealing directions for future work.

Furthermore, many natural variants of the secretary problem remain to be studied. How general a class of set systems admits a constant-factor (or even a e) approximation in the random ordering model? An appealing conjecture of Babaioff et al. [2] states that a e approximation should be possible for all matroids. We have shown that there is an interesting class of non-matroid domains - knapsack secretary problems - that admits a constant-factor approximation. Are there other natural classes of non-matroid domains that admit a constant-factor approximation?

An interesting question is how the random ordering model relates with other models of stochastic optimization. In particular, the "sample-and-optimize" approach taken in all algorithms in this paper bears superficial similarity to the standard techniques in multi-stage stochastic optimization. It would be interesting to formalize this similarity, and perhaps derive new insights into both classes of problems.

References

1. G. Aggarwal and J. Hartline. Knapsack auctions. In *SODA*, pages 1083–1092, 2006.

2. M. Babaioff, N. Immorlica, and R. Kleinberg. Matroids, secretary problems, and online mechanisms. In *SODA*, pages 434–443, 2007.
3. C. Borgs, J. Chayes, O. Etesami, N. Immorlica, K. Jain, and M. Mahdian. Dynamics of bid optimization in online advertisement auctions. In *Proceedings of the 16th International World Wide Web Conference*, 2007. to appear.
4. N. Buchbinder and J. Naor. Online primal-dual algorithms for covering and packing problems. In *ESA*, 2005.
5. N. Buchbinder and J. Naor. Improved bounds for online routing and packing via a primal-dual approach. In *FOCS*, 2006.
6. D. Chakrabarty, Y. Zhou, and R. Lukose. Budget constrained bidding in keyword auctions and online knapsack problems. In *WWW2007, Workshop on Sponsored Search Auctions*, 2007.
7. B. Dean, M. Goemans, and J. Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. In *FOCS*, pages 208–217, 2004.
8. B. Dean, M. Goemans, and J. Vondrák. Adaptivity and approximation for stochastic packing problems. In *SODA*, pages 395–404, 2005.
9. E. B. Dynkin. The optimum choice of the instant for stopping a Markov process. *Sov. Math. Dokl.*, 4, 1963.
10. J. Feldman, S. Muthukrishnan, M. Pal, and C. Stein. Budget optimization in search-based advertising auctions. In *Proceedings of the 8th ACM Conference on Electronic Commerce*, 2007. to appear.
11. R. Kleinberg. A multiple-choice secretary problem with applications to online auctions. In *SODA*, pages 630–631, 2005.
12. G. Lueker. Average-case analysis of off-line and on-line knapsack problems. In *SODA*, pages 179–188, 1995.
13. A. Marchetti-Spaccamela and C. Vercellis. Stochastic on-line knapsack problems. *Mathematical Programming*, 68:73–104, 1995.
14. R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
15. S. Muthukrishnan, M. Pal, and Z. Svitkina. Stochastic models for budget optimization in search-based. manuscript, 2007.
16. P. Rusmevichientong and D.P. Williamson. An adaptive algorithm for selecting profitable keywords for search-based advertising services. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, 2006.