

Knapsack Secretary Through Boosting*

Andreas Abels[†] Leon Ladewig[‡] Kevin Schewior[§] Moritz Stinzendörfer[¶]

August 11, 2022

Abstract

We revisit the knapsack-secretary problem (Babaioff et al.; APPROX 2007), a generalization of the classic secretary problem in which items have different sizes and multiple items may be selected if their total size does not exceed the capacity B of a knapsack. Previous works show competitive ratios of $1/(10e)$ (Babaioff et al.), $1/8.06$ (Kesselheim et al.; STOC 2014), and $1/6.65$ (Albers, Khan, and Ladewig; APPROX 2019) for the general problem but no definitive answers for the achievable competitive ratio; the best known impossibility remains $1/e$ as inherited from the classic secretary problem. In an effort to make more qualitative progress, we take an orthogonal approach and give definitive answers for special cases.

Our main result is on the 1-2-knapsack secretary problem, the special case in which $B = 2$ and all items have sizes 1 or 2, arguably the simplest meaningful generalization of the secretary problem towards the knapsack secretary problem. Our algorithm is simple: It *boosts* the value of size-1 items by a factor $\alpha > 1$ and then uses the size-oblivious approach by Albers, Khan, and Ladewig. We show by a nontrivial analysis that this algorithm achieves a competitive ratio of $1/e$ if and only if $1.40 \lesssim \alpha \leq e/(e-1) \approx 1.58$.

Towards understanding the general case, we then consider the case when sizes are 1 and B , and B is large. While it remains unclear if $1/e$ can be achieved in that case, we show that algorithms based only on the relative ranks of the item values can achieve precisely a competitive ratio of $1/(e+1)$. To show the impossibility, we use a non-trivial generalization of the factor-revealing linear program for the secretary problem (Buchbinder, Jain, and Singh; IPCO 2010).

1 Introduction

In the classic secretary problem, there is a single position to be filled, and n candidates arrive one by one in uniformly random order. Upon arrival of any candidate, they have to be rejected or accepted immediately and irrevocably only based on *ordinal* information on the candidates seen so far, that is, their relative ranks. The goal is to maximize the probability that the best candidate is selected. The origin of this problem is unclear; for a discussion, we refer to Ferguson’s survey [16]. It is well known [26, 13] since the 1960s that a probability of $1/e$ can be achieved by selecting the first candidate that is better than the n/e first candidates and that this is the best-possible probability under the typical assumption $n \rightarrow \infty$. Many extensions of this problem have since been considered, especially in recent years, partially due to relations to beyond-the-worst-case analyses of online algorithms (e.g., [20, 1, 18]) and to mechanism design (e.g., [24, 4]).

There is extensive work on multiple-choice variants of the secretary problem. Few of these works consider an ordinal setting [8, 19, 29]; the majority considers the *value* setting in which each arriving candidate (or item) i is revealed along with a value $v_i \in \mathbb{R}_{\geq 0}$ and must be rejected or accepted immediately and irrevocably so that the set of accepted items obeys some combinatorial constraint. The goal is to obtain an algorithm with a (strong) competitive ratio ρ , i.e., that constructs a solution ALG such that $v(\text{ALG})$, the sum of values of accepted items, is in expectation at least $\rho \cdot v(\text{OPT})$ where OPT is the best solution that could have been constructed.

Whereas the results for the standard secretary problem carry over to the value setting, even relatively simple variants are not completely understood in that setting. This is arguably due to the sheer amount of conceivable strategies. For instance, the precise competitive ratio achievable in the 2-secretary problem, the variant in which two positions are to be filled, is *not* known—only that it is strictly larger than in the much-better-understood ordinal “counterpart”, sometimes called the (2, 2)-secretary problem [8, 9].

The secretary variant that has probably received most attention is the matroid secretary problem [5], an extension of the k -secretary problem [24] (in which k positions are to be filled) to any matroid constraint, see, e.g.,

*Supported in part by the Independent Research Fund Denmark, Natural Sciences, grant DFF-0135-00018B.

[†]School of Business and Economics, RWTH Aachen University, Germany. Email: andreas.abels@oms.rwth-aachen.de.

[‡]Munich, Germany. Email: leonladewig@mail.de.

[§]Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark. Email: kevs@sdu.dk.

[¶]Department of Mathematics, TU Kaiserslautern, Germany. Email: stinzenendoerfer@mathematik.uni-kl.de.

the state-of-the-art result [25, 15] and the survey by Dinitz [12]. An orthogonal and also well-known extension of k -secretary is the *knapsack* secretary problem in which items additionally have sizes and the total size of accepted items must not exceed some given capacity B [4, 23, 2, 28, 21]. While this line of work has improved the competitive ratio from $1/(10e)$ to $1/6.65$, no impossibility beyond $1/e$ has been found. For some secretary versions, e.g., the bipartite-matching variant [22], it is known that this ratio can in fact be matched.

Our paper may raise hope that the ratio of $1/e$ can in fact be matched for knapsack secretary. First, we consider the 1-2-knapsack problem. Here, items have sizes either 1 or 2 and the capacity B is 2. We develop a $1/e$ -competitive algorithm. To us, this result is both surprising and significant because the problem generalizes both the classic secretary problem, which severely restricts the set of candidate algorithms, and the not-entirely-understood 2-secretary problem. We also consider the problem with sizes either 1 or B and B large, for which we show initial results, namely that $1/(e+1) \pm o(1)$ is precisely the competitive ratio that can be achieved by *ordinal* algorithms. These are algorithms that only use the relative rank of the items and disregard the actual values.

1.1 Related Work

Kleinberg [24] first considers k -secretary as introduced above, gives an algorithm with competitive ratio $1 - \Theta(1/\sqrt{k})$, and shows that this ratio is asymptotically best possible. This result is reproduced by Kesselheim et al. [23] in the more general context of packing LPs. Buchbinder et al. [8] consider the (j, k) -secretary problem in the ordinal setting in which j items can be selected and the goal is to maximize the expected ratio of elements selected from the top k items. They also state the algorithm-design problems as linear programs, which they can only solve for small values of j and k , but Chan et al. [9] can solve them for larger values. Any guarantee for the (k, k) -secretary problem carries over to the k -secretary problem, but Chan et al. [9] rule out the other direction. More specifically, Chan et al.'s results include an optimal algorithm for $(2, 2)$ -secretary with guarantee approximately 0.489 and a (not necessarily optimal) algorithm for 2-secretary with guarantee approximately 0.492. Albers and Ladewig [3] revisit the problem and give simple algorithms with improved (albeit non-optimal) competitive ratios for many fixed values of k .

The knapsack secretary problem is introduced by Babaioff et al. [4] who give a $1/(10e)$ -competitive algorithm, which was subsequently improved by Kesselheim et al. [23] to $1/8.06$ and by Albers, Khan, and Ladewig [2] to $1/6.65$. Essentially all known $\Omega(1)$ -competitive algorithms for the knapsack secretary problem are somewhat wasteful in the competitive ratio, presumably at least partially for the sake of a simpler analysis, in that they randomize between different algorithms that are tailored to respective item sizes. It seems that qualitative progress can only be made by a more fine-grained analysis avoiding such case distinctions.

A variant of the knapsack secretary problem that has recently been considered is the fractional variant in which an item can also be packed fractionally, avoiding situations in which an arriving item cannot be selected at all, even when there is space. The currently best known achievable competitive ratio is $1/4.39$ [17], also achieved by a blended approach.

It is not difficult to see that no constant competitive ratio can be achieved when the items do not arrive in random but in adversarial order, even in the unit-value case [27]. Starting from this problem, problems in which other assumptions than the order are relaxed are considered as well. For instance, Zhou et al. [30] consider the version in which each item has a small size; Böckenhauer et al. [6] and Boyar et al. [7] introduce advice and untrusted predictions, respectively, to the problem.

Lower bounds for secretary problems in the value setting are rare. For some related problems [10, 11, 14], the rich class of strategies can be handled by, for any strategy, identifying an infinite set of values (using Ramsey theory) on which it is much better behaved. It is, however, not clear how such an approach could be applied, e.g., for knapsack secretary since it seems one would need to control how the values in the support are spread out, a property that is irrelevant in the other settings.

1.2 Our Contribution

The special case 1-2-knapsack is not only arguably the simplest special case that exhibits features of the knapsack problem distinguishing it from the matroid secretary problem. Since the problem generalizes both the standard secretary problem and 2-secretary, we believe that settling it in terms of the achievable competitive ratio is also interesting per se.

A good starting point for tackling 1-2-knapsack seems to be the extended secretary algorithm, which is $1/3.08$ -competitive in the slightly more general case when all items have size larger than $B/3$ [2]. This algorithm simply ignores the item sizes, samples some prefix of length cn for some optimized constant $c \in (0, 1)$, and afterwards selects all items that surpass the largest value from the sampling phase and that can still be feasibly

packed. It is, however, easy to see that this approach cannot achieve $1/e$: Achieving $1/e$ in an instance where the optimal solution consists of a large item requires setting $c = 1/e \pm o(1)$. The resulting algorithm will, however, not be $1/e$ -competitive in an instance where the optimal solution consists of two small items of equal value, but there are many large items, each slightly more valuable than the individual small items, making sure that the small items are (almost) never selected by the algorithm. In this case, the competitive ratio of the algorithm will be essentially half the probability that the algorithm selects a (large) item, that is, $(1 - 1/e)/2 < 1/e$. We denote two instances of the above forms by \mathcal{I}_1 and \mathcal{I}_2 , respectively, in the following. Clearly, it is possible to choose c so as to balance between \mathcal{I}_1 and \mathcal{I}_2 . As a small side result, we show that a ratio of approximately $0.353 < 1/e$ can be achieved that way.

The key observation leading to our $1/e$ -competitive algorithm is that keeping $c = 1/e$ and internally multiplying (*boosting*) values of small items with a suitable constant factor $\alpha > 1$ prior to running the extended secretary algorithm may handle both \mathcal{I}_1 and \mathcal{I}_2 : While this is clear for \mathcal{I}_1 when the ranking of values does not change through boosting, a small item may overtake the most valuable (large) item. This however means that this small item has relatively large (actual) value. Using that the algorithm also accepts the second-best item with a significant probability ($1/e^2$), we can show that, with the right choice of α , we still extract enough value from the small and large items to cover $1/e \cdot v(\text{OPT})$. In \mathcal{I}_2 , the small items would overtake the large items, significantly improving the expected value achieved by the algorithm; conversely, if they did not overtake, they would not have been harmfully valuable in the first place—again with the right choice of α . To sum up, “ \mathcal{I}_1 type” instances impose an upper bound on α , and “ \mathcal{I}_2 type” instances impose a lower bound on α . We show that the algorithm is $1/e$ -competitive if *and only if* $1.40 \lesssim \alpha \leq e/(e-1) \approx 1.58$ where the upper bound comes essentially from the above consideration for \mathcal{I}_1 . Note that therefore, in particular, our boosting *is* different from ordering the items by their “bang for the buck” ratios.

We note that, while α -boosting seems reminiscent of β -filtering [9] (for $\beta < 1$), applying β -filtering to the extended secretary algorithm will not yield a $1/e$ -competitive algorithm. The extended secretary algorithm would be adapted by ignoring items with a value less than β times the highest value seen so far. Note that indeed, a “ \mathcal{I}_1 type” instance where all but the most valuable item have a similar small value, one would have to choose $c = 1/e \pm o(1)$ again, independently of β . But such an algorithm would again only be $(1 - 1/e)/2$ -competitive on \mathcal{I}_2 .

The crux of our analysis is distinguishing all possible cases beyond those covered by \mathcal{I}_1 and \mathcal{I}_2 in a smart way. To bound the algorithm’s value in each of these cases, we precisely characterize the probabilities with which the algorithm selects an item depending on its size and its position in the (boosted) order of values, significantly extending observations made by Albers and Ladewig [3].

Before tackling the general case and understanding potentially complicated knapsack configurations, we propose considering a clean special case called 1 - B -knapsack where items have sizes either 1 or B , and B is large. One may be tempted to think that this special case is difficult in that selecting a small item early on may lead to a blocked knapsack and a horribly inefficient use of capacity, e.g., because all other items are large. On the other hand, when B is large, one can easily avoid such situations by sampling. We do not give a conclusive answer on whether $1/e$ can be matched in this case, but we give some preliminary results.

Unfortunately, a competitive ratio of $1/e$ for 1 - B -knapsack cannot be achieved with our boosting approach. The same consideration we made for \mathcal{I}_1 earlier (for 1 - 2 -knapsack) to get an upper bound of $e/(e-1)$ on α still works; in contrast, a generalization of \mathcal{I}_2 rules out any constant boosting factor.

We then give another algorithm for 1 - B knapsack which can be viewed as a linear interpolation between the classic secretary algorithm and the algorithm by Kleinberg [24] for k -secretary. We show that it is $1/(e+1)$ -competitive. This algorithm turns out to be *ordinal*, that is, its decisions only depend on the item sizes and the relative order of their values. Remarkably, we are able to show that $1/(e+1)$ is the best-possible guarantee such algorithms can achieve. We do so by generalizing the factor-revealing linear program due to Buchbinder et al. [8] by adding variables and constraints. Arguing that the LP indeed models our problem becomes more difficult because, in contrast to the setting of Buchbinder et al., at any time, even the size of the next item is random. We do so by showing reductions between our model and an auxiliary batched-arrival model.

2 Preliminaries

We use the following notation. Let $\mathcal{I} = \{1, \dots, n\}$ be the set of items (also called *elements*), where each item $i \in \mathcal{I}$ is specified by a profit v_i and a size s_i . Moreover, we are given a knapsack of capacity $B \in \mathbb{N}_{\geq 2}$. The goal is to find a maximum-profit packing, i.e., a subset of items S such that $\sum_{i \in S} s_i \leq B$ and $\sum_{i \in S} v_i$ is maximized. Without loss of generality, we assume that all elements have distinct values and that $v_1 > v_2 > \dots > v_n$. This way, the name of an item i corresponds to the (global) *rank* in \mathcal{I} .

Algorithm 1: Extended secretary algorithm

Input: Instance of 1- B -knapsack arriving in uniformly random order, parameter $c \in (0, 1)$.
Output: A knapsack packing.
for round $\ell = 1$ **to** n **do**
 if $\ell \leq cn$ **then**
 Reject the current item; // sampling phase
 end
 if $\ell > cn$ **then**
 Let v^* be the highest profit seen up to round $\lfloor cn \rfloor$;
 Pack the current item if its profit exceeds v^* and the remaining capacity is large enough;
 end
end

Throughout the following sections, an important subclass of the knapsack problem arises where each item has either size 1 or B .

Definition 1 (1- B -knapsack). *We call the special case of the knapsack problem where all items have size 1 or B the 1- B -knapsack problem. Items of size 1 are called small and items of size B are called large.*

Within the context of 1- B -knapsack, we use the following further notation. Let \mathcal{I}_S be the set of small items. For any small item $i \in \mathcal{I}_S$, let $r_s(i)$ denote its rank among the small items. Note that $r_s(i)$ is at most the global rank i of this item. Further, let $r'_g(a)$ denote the global rank of the small item x that satisfies $r_s(x) = a$. When we use just the word “rank”, we refer to the global rank.

Let **OPT** be an optimal offline algorithm. For any algorithm **ALG**, we overload the notation and use the same symbol also for the packing returned by the algorithm. Further, we denote by $v(\text{ALG}) := \sum_{i \in \text{ALG}} v_i$ the total profit of the packing returned by **ALG**. We are particularly interested in *online* algorithms, i.e., algorithms that are initially only given n and are presented with the items one by one. Upon arrival of any item, an online algorithm has to irrevocably decide whether it includes the item or not. A special class of algorithms we consider are *ordinal* algorithms. These algorithms only have access to the item sizes and the *relative order* of item values.

We say that an online algorithm **ALG** is ρ -competitive if $\mathbb{E}[v(\text{ALG})] \geq \rho \cdot v(\text{OPT})$ for all instances, where the expectation is taken over a uniformly random arrival order (and possibly internal randomization that the algorithm uses). In general, we assume $n \rightarrow \infty$ for our bounds. Note that, for a fixed number of items, we can achieve a guarantee that is arbitrarily close to the guarantee for $n \rightarrow \infty$ by adding a sufficient amount of virtual dummy items.

Finally, throughout the paper, we use the notation $[k] := \{1, \dots, k\}$ for any $k \in \mathbb{N}$.

3 Matching $1/e$ for 1-2-Knapsack

In this section, we develop an optimal algorithm for 1-2-knapsack. For this purpose, we first propose a natural algorithm for 1- B -knapsack, based on the size-oblivious approach from [2]. Here, items are accepted whenever their profit exceeds a certain threshold, similar to the optimal algorithm for the classic secretary problem. Therefore, we call it the *extended secretary algorithm*. From an initial sampling phase of length cn , where $c \in (0, 1)$ is a parameter of the algorithm, the best item is used as a reference element. Subsequently, any item beating the reference element is packed if it still fits. A formal description is given in Algorithm 1.

In the following, we denote Algorithm 1 by **ALG** and set

$$\begin{aligned} p_i(j) &:= \Pr[\text{ALG packs item } i \text{ as the } j\text{-th element}], \\ p_i &:= p_i(1), \\ P_i &:= \sum_{j=1}^B p_i(j). \end{aligned}$$

Thus, P_i is the probability that the algorithm packs item i at all, while p_i is the probability that it is packed as the first item. We first state some results on the values p_i , which have essentially been investigated in [3]. Indeed, the following results follow from that work and some simple observations.

Lemma 1. *For $i \in \mathbb{N}$, it holds that*

$$p_i = c \left(\ln \frac{1}{c} + \sum_{\ell=1}^{i-1} (-1)^{\ell+1} \binom{i-1}{\ell} \frac{c^\ell - 1}{\ell} \right) \pm o(1).$$

Proof. Let $i \in \mathbb{N}$. The extended secretary algorithm packs i as the first item if and only if the SINGLE-REF algorithm from [3] with $r = 1$ and $k = i$ packs i as the first item. Hence, the probability p_i can be derived from [3] as follows: If $i = 1$, item i is a dominating item in the terminology of [3] and Lemma 6 of [3] gives $p_1 = c \cdot \ln(1/c) - o(1)$. In the case $i \geq 2$, item i is a non-dominating item in the terminology of [3]. Here, Lemma 4 of [3] gives $p_i = p_i(i)$ and Lemma 5 of [3] and gives $p_i(i) = p_1(i)$, that is, p_i turns out to be the probability that the dominating item 1 is accepted as the i -th item by the SINGLE-REF algorithm. Again, the claim follows from Lemma 6 of [3]. \square

Furthermore, observe that, since increasing the profit of an item cannot decrease its probability of being selected, we have $p_i \geq p_{i+1}$ for all $i \in [n-1]$. Note that ALG accepts no item if and only if the best item is in the sampling phase. Therefore, we have the following observation.

Observation 1. *It holds that*

$$\begin{aligned} \sum_{i=1}^n p_i &= 1 - \Pr[\text{ALG accepts no item}] \\ &= 1 - \Pr[\text{item 1 appears in sampling phase}] = 1 - c. \end{aligned}$$

In the following subsection, we identify relations between the probabilities P_i and p_j .

3.1 Structural Lemma

In this subsection, we show the following lemma connecting the probabilities P_i to the probabilities p_j from Lemma 1. The analysis showing the $1/e$ -competitiveness of our algorithm is crucially based on this result. Note that we only use it for $B = 2$ but it holds for all B .

Lemma 2. *The probability that ALG packs element $i \in \mathcal{I}$ is*

$$P_i = \begin{cases} p_i & \text{if element } i \text{ is large,} \\ i_s^* \cdot p_i + \sum_{x=r_s(i)+1}^{B^*} p_{r'_s(x)} & \text{if element } i \text{ is small,} \end{cases} \quad (1)$$

$$(2)$$

with $i_s^* := \min\{r_s(i), B\}$ and $B^* := \min\{B, |\mathcal{I}_S|\}$.

Observe that (1) follows immediately: Any large element can only be packed when the knapsack is empty, i.e., as the first element. The proof of (2) requires a bit more work.

Definition 2. Let $E_{x,y}^{i,j}$ be the event that the small elements i and j are packed as the x -th and y -th items, respectively.

Note that the event that any item $i \in \mathcal{I}_S$ is packed as x -th item, where $x \geq 2$, can be partitioned according to the item packed first. Therefore, for any $i \in \mathcal{I}_S$ and $x \geq 2$,

$$p_i(x) = \sum_{j \in \mathcal{I}_S} \Pr \left[E_{1,x}^{j,i} \right]. \quad (3)$$

We have the following technical lemmata.

Lemma 3. *Let $i \in \mathcal{I}_S$ be any small item and $i_s^* = \min\{r_s(i), B\}$. For $2 \leq \ell \leq i_s^*$, it holds that $\sum_{j \in \mathcal{I}_S} \Pr \left[E_{1,\ell}^{i,j} \right] = p_i$.*

Proof. The first step is to show that at least ℓ elements are accepted in total, if element i is accepted first. Since element i has rank $r_s(i)$ among the small elements, there are $r_s(i) - 1$ small elements that are more valuable. Their position in the input sequence cannot be in the sampling phase, nor before element i if it is packed first. So there are at least i_s^* small elements that can be packed subsequently. Therefore, for $2 \leq \ell \leq i_s^*$, a small element is packed as ℓ -th item. The claim follows by partitioning the event that i is packed first according to the item $j \in \mathcal{I}_S$ packed as ℓ -th item. \square

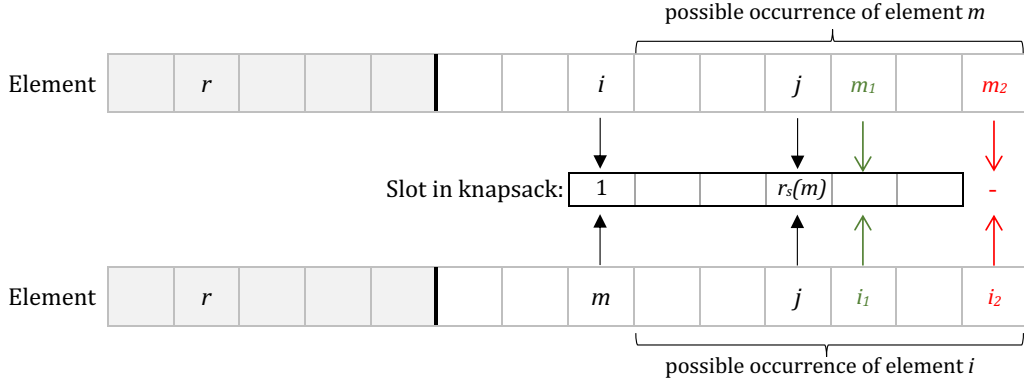


Figure 1: Occurrence of element i and m in event $E_{1,r_s(m)}^{m,j}$ and $E_{1,r_s(m)}^{i,j}$

Lemma 4. For any two small elements $i, j \in \mathcal{I}_S$ and any $x, y \in [B]$, we have $\Pr[E_{x,y}^{i,j}] = \Pr[E_{x,y}^{j,i}]$.

Proof. Consider any input sequence of $E_{x,y}^{i,j}$ and the sequence resulting from swapping the elements i and j . Since both elements are not part of the sample, the reference element is not changed by the swap. Therefore, no element that was previously accepted will be rejected and none that was previously rejected will be accepted. Only the order of selection changes. \square

Lemma 5. For any small items $i, j, m \in \mathcal{I}_S$ with $r_s(m) > 1$ and $r_s(i) < r_s(m)$, it holds that $\Pr[E_{1,r_s(m)}^{m,j}] = \Pr[E_{1,r_s(m)}^{i,j}]$.

Proof. Consider any input sequence from $E_{1,r_s(m)}^{m,j}$. Since $r_s(i) < r_s(m)$ applies, element i lies behind the element with rank m in the sequence. If both are selected (see i_1 in Figure 1), this also applies after they have been swapped (see Lemma 4 and m_1 in Figure 1). If previously only element m of the two is packed, only element i (of the two) is selected after their swapping (i_2 and m_2 in Figure 1), since in this case, nothing changes in the reference element either. Therefore $\Pr[E_{1,r_s(m)}^{m,j}] \leq \Pr[E_{1,r_s(m)}^{i,j}]$ applies.

Now consider any input sequence from $E_{1,r_s(m)}^{i,j}$. We show that the element m lies behind the element i in the sequence since an element is packed as z -th item, where $z = r_s(m)$. Assuming this did not apply and m is in the sample, then there would be at most $r_s(m) - 1$ small elements that can be packed.

In the case that it occurs in the sequence after the sampling phase, but before element i , there must be a more valuable element in the sample (because m was not packed) and therefore there are again at most $r_s(m) - 1$ small elements that can be selected. In particular, in both cases, no element is packed as z -th item for $r_s(m)$. This is a contradiction to the fact that we consider an input sequence in $E_{1,r_s(m)}^{i,j}$. Now, using the same argumentation as in the first case, it follows that $\Pr[E_{1,r_s(m)}^{m,j}] \geq \Pr[E_{1,r_s(m)}^{i,j}]$, which completes the proof. \square

Using Lemmas 3 to 5, we are now able to prove Lemma 2.

Proof of Lemma 2. Let $i \in \mathcal{I}$ be any item. If i is large, it can only be packed as the first item, thus $P_i = p_i$. Now, assume that i is small. It holds that

$$P_i = \sum_{x=1}^{B^*} p_i(x) = \underbrace{\sum_{x=1}^{i_s^*} p_i(x)}_{(*)} + \underbrace{\sum_{x=r_s(i)+1}^{B^*} p_i(x)}_{(**)}.$$

We next simplify both starred terms using Lemmas 3 to 5. For $(*)$, it holds that

$$\sum_{x=1}^{i_s^*} p_i(x) = p_i(1) + \sum_{x=2}^{i_s^*} \sum_{j \in \mathcal{I}_S} \Pr[E_{1,x}^{j,i}] \quad (\text{Equation (3)})$$

$$= p_i + \sum_{x=2}^{i_s^*} \sum_{j \in \mathcal{I}_S} \Pr[E_{1,x}^{i,j}] \quad (\text{Lemma 4})$$

$$\begin{aligned}
&= p_i + \sum_{x=2}^{i_s^*} p_i(1) && \text{(Lemma 3)} \\
&= i_s^* \cdot p_i.
\end{aligned}$$

For (**), we obtain

$$\begin{aligned}
\sum_{x=r_s(i)+1}^{B^*} p_i(x) &= \sum_{x=r_s(i)+1}^{B^*} \sum_{j \in \mathcal{I}_S} \Pr \left[E_{1,x}^{j,i} \right] && \text{(Equation (3))} \\
&= \sum_{x=r_s(i)+1}^{B^*} \sum_{j \in \mathcal{I}_S} \Pr \left[E_{1,x}^{i,j} \right] && \text{(Lemma 4)} \\
&= \sum_{x=r_s(i)+1}^{B^*} \sum_{j \in \mathcal{I}_S} \Pr \left[E_{1,x}^{r'_g(x),j} \right] && \text{(Lemma 5, } r_s(i) < x) \\
&= \sum_{x=r_s(i)+1}^{B^*} p_{r'_g(x)}, && \text{(Lemma 3, } 2 \leq x = r_s(r'_g(x)) \leq B)
\end{aligned}$$

which completes the proof. \square

The following corollary is an immediate consequence of Lemma 2 for $B = 2$.

Corollary 1. *For $B = 2$, the probability that ALG packs element $i \in \mathcal{I}$ is*

$$P_i = \begin{cases} p_i & \text{if } i \text{ is large,} \\ p_i + p_{r'_g(2)} & \text{if } i \text{ is small and } r_s(i) = 1, \\ 2p_i & \text{if } i \text{ is small and } r_s(i) > 1, \end{cases}$$

where, if the second most valuable small item does not exist, we set $p_{r'_g(2)} = 0$.

3.2 First approach: Without Boosting

In this subsection, we study Algorithm 1 (as is) for 1-2-knapsack. Unfortunately, there are two instances such that it is impossible to choose the parameter c so that Algorithm 1 is $(1/e)$ -competitive on both instances.

Lemma 6. *For 1-2-knapsack, the competitive ratio of ALG is at most 0.35767, assuming $n \rightarrow \infty$.*

Proof. Let $1 > \varepsilon > 0$ be a constant. We define two instances \mathcal{I}_1 and \mathcal{I}_2 . In the first instance \mathcal{I}_1 , all items are large and only one item has substantial profit. Formally, let $v_1 = 1$, $v_i = \varepsilon^i$ for $2 \leq i \leq n$, and $s_i = 2$ for all $1 \leq i \leq n$. Then, for instance \mathcal{I}_1 ,

$$\lim_{\varepsilon \rightarrow 0} \mathbb{E}[v(\text{ALG})] = P_1 \cdot v_1 = p_1 \cdot v(\text{OPT}). \quad (4)$$

In the second instance \mathcal{I}_2 , most items are large and essentially of the same profit. However, the optimal packing contains two small items that appear at ranks $n-1$ and n . Formally, set $s_i = 2$ for $1 \leq i \leq n-2$, $s_{n-1} = s_n = 1$, and $v_i = 1 + \varepsilon^i$ for all $i \in \{1, \dots, n\}$. As item n never beats any reference item, we have $P_n = 0$. Hence, the algorithm selects only items from $\{1, \dots, n-1\}$ with positive probability, and always at most one item. For instance \mathcal{I}_2 , we get

$$\begin{aligned}
\lim_{\varepsilon \rightarrow 0} \mathbb{E}[v(\text{ALG})] &= \lim_{\varepsilon \rightarrow 0} \sum_{i=1}^n (P_i \cdot (1 + \varepsilon^i)) = \sum_{i=1}^n p_i \\
&\stackrel{\text{Obs. (1)}}{=} 1 - c \leq \frac{1-c}{2} \cdot v(\text{OPT}).
\end{aligned} \quad (5)$$

Overall, by Equations (4) and (5), the competitive ratio as $n \rightarrow \infty$ of ALG is bounded from above by

$$\max_{c \in (0,1)} \min \left\{ p_1, \frac{1-c}{2} \right\} = \max_{c \in (0,1)} \min \left\{ c \cdot \ln \frac{1}{c}, \frac{1-c}{2} \right\} \leq 0.35767.$$

This completes the proof. \square

As a small side result, we show that this bound is almost tight. The techniques are similar to those used for our main result and presented in the full version of the paper.

Proposition 1. *For 1-2-knapsack, the competitive ratio of ALG is $0.35317 - o(1)$, setting $c = 0.26888$ and assuming $n \rightarrow \infty$.*

3.3 Optimal algorithm through α -Boosting

The proof of Lemma 6 reveals the bottleneck of Algorithm 1: If the optimal solution consists of two elements having a high rank, the probability of selecting those items is small. This problem can be resolved by the concept of α -boosting.

Definition 3 (α -boosting). *Let $\alpha \geq 1$ be the boosting factor. For any item $i \in \mathcal{I}$, we define its boosted profit to be*

$$v'_i = \begin{cases} \alpha \cdot v_i & \text{if } i \text{ is small,} \\ v_i & \text{otherwise.} \end{cases}$$

In the following, we investigate Algorithm 1 enhanced by the concept of α -boosting, denoted by ALG_α . This algorithm works exactly as given in the description of Algorithm 1, but works with the boosted profit v'_i instead of the actual profit v_i for any item $i \in \mathcal{I}$. Note that the unboosted algorithm analyzed in Proposition 1 is ALG_1 . For the remainder of this subsection, we fix $c = 1/e$. In particular, this implies $p_1 = 1/e \pm o(1)$ and $p_2 = 1/e^2 \pm o(1)$ according to Lemma 1.

So far, we did not specify the boosting factor α . However, the following intuitive reasoning already shows that α should be bounded from above and below: If α is too large, we risk that ALG_α packs small items with high probability, even when they are not part of the optimal packing. On the other hand, by the result of Proposition 1 we know that ALG_1 cannot achieve an optimal competitive ratio. The following theorem provides lower and upper bounds on α such that ALG_α is $(1/e)$ -competitive.

Theorem 1. *For 1-2-knapsack, algorithm ALG_α is $(1/e - o(1))$ -competitive if and only if $1.400382 \lesssim \alpha \leq e/(e-1)$ and $c = 1/e$, assuming $n \rightarrow \infty$.*

Proof. For any item $x \in \mathcal{I}$, let $\rho(x)$ denote the global rank of x after boosting. On a high level, we need to consider two cases.

In the first case, the optimal packing contains a single item x . If $\rho(x) = 1$, we immediately obtain $\mathbb{E}[v(\text{ALG}_\alpha)] \geq p_1 v_x = (1/e) \cdot v(\text{OPT})$. Now, suppose $\rho(x) \geq 2$. Let a and b be the items such that $\rho(a) = 1$ and $\rho(b) = 2$, respectively. Hence,

$$v'_a > v'_b \geq v'_x \geq v(\text{OPT}).$$

We note that a is small, as otherwise $v_a = v'_a > v(\text{OPT})$. Moreover, for $\alpha < 2$, item b is large: If b was small, it would follow that $v'_b = \alpha \cdot v_b$ and therefore $v_a + v_b = v'_a/\alpha + v'_b/\alpha > (2/\alpha) \cdot v(\text{OPT}) > v(\text{OPT})$, contradicting the assumption that the optimal packing contains a single item. Therefore, a is small and b is large, implying $v_a = v'_a/\alpha > v(\text{OPT})/\alpha$ and $v_b = v'_b \geq v(\text{OPT})$. Hence,

$$\begin{aligned} \mathbb{E}[v(\text{ALG}_\alpha)] &\geq p_1 \cdot v_a + p_2 \cdot v_b \\ &= \left(\frac{1}{e} \pm o(1)\right) \cdot \frac{v(\text{OPT})}{\alpha} + \left(\frac{1}{e^2} \pm o(1)\right) \cdot v(\text{OPT}) \\ &\geq \left(\frac{1}{e} \pm o(1)\right) \cdot v(\text{OPT}), \end{aligned} \tag{6}$$

where the latter inequality holds for $\alpha \leq e/(e-1)$. Note that, when $v'_a = 1$, $v'_b = 1 - \varepsilon$, and $v'_z = O(\varepsilon)$ for all other items y , Inequality (6) becomes satisfied with equality as $\varepsilon \rightarrow 0$. Therefore, ALG_α is not $(1/e - o(1))$ -competitive when $\alpha > e/(e-1)$.

In the remainder of the proof, we consider the case where the optimal packing contains two small items x and y , where we assume $v_x > v_y$ without loss of generality. We set $j := \rho(x)$ and $k := \rho(y)$, where $1 \leq j < k$. Now, let a_1, \dots, a_{j-1} and b_{j+1}, \dots, b_{k-1} denote the items appearing before x and between x and y , respectively, in the ordered sequence of boosted profits:

$$v'_{a_1} > \dots > v'_{a_{j-1}} > v'_x > v'_{b_{j+1}} > \dots > v'_{b_{k-1}} > v'_y.$$

We observe that neither a items nor b items can be small: Otherwise, the profit of such an item would be strictly larger than v_y , and as any two small items fit together, this item should be in the optimal packing instead

Table 1: Upper bounds on $\theta_{j,k}$ for $3 \leq k \leq 10$ according to Equation (9).

k	3	4	5	6	7	8	9	10
$\frac{1/e - 3p_k}{\sum_{i=2}^{k-1} p_i}$	1.3475	1.3962	1.400382	1.3988	1.3968	1.3952	1.3941	1.3934

of y . Therefore, we have $v_{a_i} = v'_{a_i} > v'_x = \alpha \cdot v_x$ for all $i \in \{1, \dots, j-1\}$ and $v_{b_i} = v'_{b_i} > v'_y = \alpha \cdot v_y$ for all $i \in \{j+1, \dots, k-1\}$.

Now, we can bound the expected profit of ALG_α as follows:

$$\begin{aligned}
 \mathbb{E}[v(\text{ALG}_\alpha)] &\geq \left(\sum_{i=1}^{j-1} P_i \cdot \alpha \cdot v_x \right) + P_j \cdot v_x + \left(\sum_{i=j+1}^{k-1} P_i \cdot \alpha \cdot v_y \right) + P_k \cdot v_y \\
 &= \left(\sum_{i=1}^{j-1} p_i \cdot \alpha \cdot v_x \right) + (p_j + p_k) \cdot v_x + \left(\sum_{i=j+1}^{k-1} p_i \cdot \alpha \cdot v_y \right) + 2p_k \cdot v_y \\
 &= \underbrace{\left(p_j + p_k + \alpha \cdot \sum_{i=1}^{j-1} p_i \right)}_{\lambda_x} \cdot v_x + \underbrace{\left(2p_k + \alpha \cdot \sum_{i=j+1}^{k-1} p_i \right)}_{\lambda_y} \cdot v_y,
 \end{aligned} \tag{7}$$

where we use Corollary 1 for the first equality.

If $\lambda_x < \lambda_y$ we immediately get $\lambda_x v_x + \lambda_y v_y > \lambda_x (v_x + v_y) \geq p_1 (v_x + v_y) = (1/e) \cdot v(\text{OPT})$. Therefore, we assume $\lambda_x \geq \lambda_y$ in the following. By Chebyshev's sum inequality, it holds that $\lambda_x v_x + \lambda_y v_y \geq (1/2) \cdot (\lambda_x + \lambda_y) \cdot (v_x + v_y)$. Therefore, the competitive ratio is

$$\frac{\mathbb{E}[v(\text{ALG}_\alpha)]}{v(\text{OPT})} \geq \frac{\lambda_x + \lambda_y}{2} = \frac{1}{2} \cdot \left((1-\alpha) \cdot p_j + 3p_k + \alpha \cdot \sum_{i=1}^{k-1} p_i \right). \tag{8}$$

If $k = 2$, it follows that $j = 1$ and therefore Equation (8) resolves to

$$\mathbb{E}[v(\text{ALG}_\alpha)] \geq \frac{1}{2} \cdot (p_1 + 3p_2) \cdot v(\text{OPT}) > \frac{1}{e} \cdot v(\text{OPT}),$$

which holds independently of α . For $k \geq 3$, ALG_α is $(1/e - o(1))$ -competitive by Equation (8) if

$$\alpha \geq \frac{2/e - p_j - 3p_k}{\sum_{i=1}^{k-1} p_i - p_j} =: \theta_{j,k}.$$

It remains to show $\theta_{j,k} \leq 1.400382$ for all $k \geq 3$ and j with $1 \leq j < k$. For this purpose, we first show

$$\theta_{j,k} = \frac{2/e - p_j - 3p_k}{\sum_{i=1}^{k-1} p_i - p_j} \leq \frac{2/e - p_1 - 3p_k}{\sum_{i=1}^{k-1} p_i - p_1} = \frac{1/e - 3p_k}{\sum_{i=2}^{k-1} p_i} \pm o(1) \quad \text{for any } k \geq 3. \tag{9}$$

Since p_j is decreasing in j , the inequality in Equation (9) follows immediately if we can show $2/e - 3p_k > \sum_{i=1}^{k-1} p_i$ for large-enough n . This inequality is easily verified for $k = 3$, as $2/e - 3p_3 > p_1 + p_2$, for large-enough n . For $k \geq 4$, note that $p_k < p_1 - 1/3$, again for large-enough n , which is equivalent to $2/e - 3p_k > 1 - p_1$. Using Observation 1, we obtain $\sum_{i=1}^{k-1} p_i < \sum_{i=1}^n p_i = 1 - c = 1 - p_1$. Combining both inequalities yields Equation (9).

By computing the last term in Equation (9) for $3 \leq k \leq 10$, we obtain the upper bounds on $\theta_{j,k}$ given in Table 1, up to additive $o(1)$ terms. Note that the maximum value is 1.400382. For $k \geq 11$, we obtain from Equation (9) together with $p_i \geq 0$ for all $i \geq 11$ that

$$\theta_{j,k} \leq \frac{1/e - 3p_k}{\sum_{i=2}^{k-1} p_i} \leq \frac{1/e}{\sum_{i=2}^{11-1} p_i} < 1.398875 \pm o(1).$$

For the lower bound of approximately 1.400382 on α , first note that for $j = 1$ and $k = 5$, it holds indeed that

$$\theta_{1,5} = \frac{2/e - p_1 - 3p_5}{\sum_{i=1}^{5-1} p_i - p_1} = \frac{1/e - 3p_5}{p_2 + p_3 + p_4} \pm o(1)$$

$$= -\frac{51}{16} + \frac{9}{4e} + \frac{75 - 522e + 486e^2}{16 - 96e + 288e^2 - 64e^3} \pm o(1) \approx 1.400382 \pm o(1).$$

Next, note that setting $v'_x, v'_{b_2}, v'_{b_3}, v'_{b_4}$, and v'_y all equal to $1 + O(\varepsilon)$ and $v'(z) = O(\varepsilon)$ for all other items z makes Inequality (7) as well as Inequality (8) tight as $\varepsilon \rightarrow 0$. Therefore, the above arguments imply that $\alpha \geq \theta_{1.5}$ if and only if ALG_α is $(1/e - o(1))$ -competitive. This completes the proof. \square

4 Ordinal Algorithms for 1- B -Knapsack

In this section, we consider ordinal algorithms for 1- B -knapsack with B large. Recall that ordinal algorithms have access to both item sizes and the relative order on item values (of previously arrived items) but not to the actual item values. We show the following theorem.

Theorem 2. *There is an ordinal $(1/(e+1) - o(1))$ -competitive algorithm for the 1- B -knapsack problem, and every ordinal algorithm has a competitive ratio of at most $1/(e+1) + o(1)$ for this problem.*

We first discuss the lower bound, i.e., the algorithm. Note that, while the input is any combination of large and small items, the optimal solution still consists of either the single most valuable item OPT_L or of a set of up to B small items OPT_S . Our algorithm can be viewed as a linear combination of (near-)optimal algorithms ALG_L and ALG_S against the respective cases. In particular, ALG_L is the $(1/e)$ -competitive algorithm [16] for the standard secretary problem and run with probability $e/(e+1)$; ALG_S is the $(1 - o(1))$ -competitive algorithm for k -secretary by Kleinberg [24] and run with probability $1/(e+1)$. The competitive ratio follows by a simple case distinction. A small subtlety that we need to take care of is that these subroutines require the number of items as input. To deal with this problem, we introduce dummy items. In the following, we make this idea formal.

Proof (Algorithm). The algorithm ALG_L treats all items as if they were large and then applies the standard secretary algorithm [26, 13]. For the algorithm ALG_S , whenever a large item arrives, we pretend that a small dummy item with value 0 arrives. These dummy items can be accepted and take up space in the capacity constraint, but they do not contribute to the solution value. On this adapted instance, we apply an optimal algorithm for the multiple-choice secretary problem, e.g. Kleinberg [24] or Kesselheim et al. [23]. Clearly, for both algorithms, any solution for the respective adapted instance can be translated back to a solution with equal value for the original instance. Also, both of these algorithms are ordinal.

For every input instance, our algorithm chooses ALG_L with probability $\frac{e}{e+1}$ and ALG_S otherwise. To analyze the competitive ratio, distinguish two cases. If $\text{OPT} = \text{OPT}_L$, we use that the algorithm chooses ALG_L with probability $e/(e+1)$ and conditioned on that achieves an expected value of $v(\text{OPT}_L)/e$ [26, 13], yielding an unconditional expected value of $v(\text{OPT}_L)/(e+1)$. Otherwise, i.e., if $\text{OPT} = \text{OPT}_S$, we use that ALG_S is run with probability $1/(e+1)$ which achieves, as $B \rightarrow \infty$, an expected value of $(1 - o(1)) \cdot v(\text{OPT}_S)$, resulting in an unconditional expected value of $(1 - o(1)) \cdot v(\text{OPT}_S)/(e+1)$. \square

We now discuss the upper bound, i.e., the impossibility. In our construction, there are B large and B small items. All items have different values, and each large item is more valuable than each small item. The adversary chooses between two ways of setting the values: The first option is to make the solution consisting of *all* small items much more valuable than any single large item; the second option is to make a single large item much more valuable than any other solution.

Ideally, we would like to analyze algorithms in the following setting: In each of n rounds, the algorithm is presented with both a uniformly random small and a uniformly random large item out of the items not presented thus far. Upon presentation of any such two items, the algorithm has to choose whether to select all small items from now on or to select the current large item. While the actual setting, in which all items arrive in uniformly random order, is clearly different, we show below that working with the other setting is only with a $(1 \pm o(1))$ -factor loss in the impossibility by reductions between our problem and an auxiliary batched-arrival model.

Assuming the latter setting, we can write a linear program similar to that of Buchbinder et al. [8]. Like in that approach, each LP solution corresponds to an algorithm and vice versa. More specifically, our LP uses two variables (rather than one) for every time step, corresponding to the probabilities that the algorithm accepts a large item or the first small item, respectively. In addition, there is a variable representing the competitive ratio, and there are two upper bounds (rather than one) on that variable, representing the two instances the adversary can choose. A feasible dual solution then yields the desired impossibility. We formalize these ideas in the following.

$$\begin{array}{ll}
\max & c \\
s.t. & c \leq \frac{1}{k} \sum_{i=1}^k (i \cdot p_i) \\
& c \leq \sum_{i=1}^k \left[\left(1 - \frac{i-1}{k}\right) \cdot q_i \right] \\
& i \cdot p_i \leq 1 - \sum_{j=1}^{i-1} (p_j + q_j) \quad \forall i \in [k] \\
& q_i \leq 1 - \sum_{j=1}^{i-1} (p_j + q_j) \quad \forall i \in [k] \\
& p_i, q_i \geq 0 \quad \forall i \in [k]
\end{array}
\qquad
\begin{array}{ll}
\min & \sum_{i=1}^k (x_i + y_i) \\
s.t. & \alpha + \beta = 1 \\
& i \cdot x_i + \sum_{j=i+1}^k (x_j + y_j) \geq \frac{i}{k} \cdot \alpha \quad \forall i \in [k] \\
& y_i + \sum_{j=i+1}^k (x_j + y_j) \geq \left(1 - \frac{i-1}{k}\right) \cdot \beta \quad \forall i \in [k] \\
& x_i, y_i \geq 0 \quad \forall i \in [k] \\
& \alpha, \beta \geq 0
\end{array}$$

Figure 2: The primal and dual linear programs used in our proof of the upper bound in Theorem 2.

Proof (Impossibility). Consider the following two instances that are treated identically by ordinal algorithms. There are $n = 2B$ items where items $i \in \{1, \dots, B\}$ are large and items $i \in \{B+1, \dots, 2B\}$ are small. In one instance, the item values are $v_i = 1 + (B-i) \cdot \varepsilon$ for $i \leq B$ and $v_i = 1 - i\varepsilon$ for $i > B$. In the other instance, the values are the same except for $v_1 = B^2$. So, for both instances, the rank of item i is indeed i , for all $i \in \{1, \dots, 2B\}$. The two optimal solutions are $\text{OPT}_L = \{1\}$ and $\text{OPT}_S = \{B+1, B+2, \dots, 2B\}$. The adversary decides which of the two instances is the actual instance.

We consider the following batched-arrival setting parameterized with some constant k and assume that k divides n . The items still arrive in uniformly random order, but the algorithm does not always have to make a decision upon the arrival of an item. More specifically, for any $i \in \{1, \dots, k\}$, upon the arrival of the $(i \cdot n/k)$ -th item, the algorithm may make a decision about all items that have arrived in the current batch, i.e., after the $((i-1) \cdot n/k)$ -th item. Clearly, any upper bound on the competitive ratio achievable in this setting, is also an upper bound on the competitive ratio achievable in the original setting.

Note that the expected number of items of each type, i.e., small and large, in each batch is $n/(2k)$. Let $\delta > 0$ be some constant. As follows from a standard concentration (e.g., Chernoff) bound, when $n \rightarrow \infty$, the probability that the number of items from each type is between $(1-\delta) \cdot n/(2k)$ and $(1+\delta) \cdot n/(2k)$ approaches 1. From the union bound over all batches it then follows that also the probability that the number of items of each type *in each batch* is within the given range approaches 1. We may therefore assume that this is indeed the case at an arbitrarily small loss in our impossibility.

To analyze the algorithm in the batched-arrival setting, we write a linear program similar to that of Buchbinder et al. [8]. The LP encodes a probability distribution for the decisions that an algorithm **ALG** makes against the pair of instances. The variable p_i represents the probability that the algorithm selects the best large item from the i -th batch. Similarly, the variable q_i represents the probability that the algorithm selects all small items from both the i -th batch and forthcoming batches.

Note that the algorithm may make any such decision, i.e., selecting the best largest item or starting to select small items from a batch, for at most a single batch. Hence, we obtain $q_i \leq 1 - \sum_{j=1}^{i-1} p_j + q_j$ as a constraint for our LP for all $i \in [k]$. Further, observe that we may assume that the algorithm only selects a large item when the best largest item so far is in the current batch. In batch i , the probability for this to happen is at most $(1+\delta)/((1+\delta) + (i-1) \cdot (1-\delta))$. As $\delta \rightarrow 0$, we obtain

$$p_i \leq \left(1 - \sum_{j=1}^{i-1} p_j + q_j\right) \cdot \frac{1}{i}$$

for all $i \in [k]$, another constraint of the LP.

The objective function of the LP is c , an upper bound on the competitive ratio of the algorithm. For each of the two instances that the adversary could choose, we write an additional constraint upper bounding c . If the adversary chooses the first instance and the algorithm starts selecting small items at the end of the i -th batch, the fraction of $v(\text{OPT}_S)$ the algorithm obtains is at most

$$\frac{(k-i+1) \cdot (1+\delta)}{(k-i+1) \cdot (1+\delta) + (i-1) \cdot (1-\delta)} \xrightarrow{\delta \rightarrow 0} 1 - \frac{i-1}{k}.$$

Hence, as $\delta \rightarrow 0$, we obtain the constraint $c \leq \sum_{i=1}^k (1 - \frac{i-1}{k}) q_i$. Now consider the case that the adversary chooses the second instance. Suppose that the algorithm selects the best large item from the i -th batch, which is by assumption the best item that has already arrived. Since the order of large items is a uniformly random order, the probability that the chosen item is the globally best large item is the fraction of already observed large items within the whole instance, that is, at most

$$\frac{i \cdot (1 + \delta)}{i \cdot (1 + \delta) + (k - i) \cdot (1 - \delta)} \xrightarrow{\delta \rightarrow 0} \frac{i}{k}.$$

Hence, as $\delta \rightarrow 0$, we get the constraint $c \leq \sum_{i=1}^k (p_i \cdot \frac{i}{k}) = \frac{1}{k} \sum_{i=1}^k i \cdot p_i$. We give both the resulting LP and its dual in Figure 2.

We give a solution to the dual LP. Let τ be the integer number such that

$$\sum_{i=\tau}^{k-1} \frac{1}{i} < 1 \leq \sum_{i=\tau-1}^{k-1} \frac{1}{i}.$$

We set $y_i = 0$ for all $i < k$, $y_k = 1/((e+1) \cdot k)$, $x_i = 0$ for $i < \tau$, and

$$x_i = \frac{e}{(e+1) \cdot k} \cdot \left(1 - \sum_{j=i}^{k-1} \frac{1}{j} \right)$$

for $i \geq \tau$. Further, $\alpha = e/(e+1)$ and $\beta = 1/(e+1)$. Note that this choice of x is analogous to the dual solution by Buchbinder et al. [8] but scaled by a factor of α .

We argue that the solution is feasible when x is scaled up by a $(1 + o(1))$ factor (where the Landau symbol is with respect to $k \rightarrow \infty$). Clearly, $\alpha + \beta = 1$. The inequality $ix_i + \sum_{j=i+1}^k x_j + y_j \geq \frac{i}{k} \cdot \alpha$ is the same as in the dual by Buchbinder et al., except for additional y variables on the left-hand side and a scaling by α of the right-hand side. Therefore with our choice of x (which is scaled up by α compared to Buchbinder et al.), the inequalities are identical and the previous proof of feasibility also holds, even without additional scaling of x . We consider the remaining (new) inequalities. For $i = 1$, we have

$$\begin{aligned} y_1 + \sum_{j=2}^k x_j + y_j &\geq \frac{e}{(e+1) \cdot k} \cdot \sum_{j=\tau}^k \left(1 - \sum_{\ell=j}^{k-1} \frac{1}{\ell} \right) \\ &= \frac{e}{(e+1) \cdot k} \cdot \left((k - \tau + 1) - \sum_{j=\tau}^k \sum_{\ell=j}^{k-1} \frac{1}{\ell} \right) \\ &= \frac{e}{(e+1) \cdot k} \cdot \left((k - \tau + 1) - \sum_{j=\tau+1}^k \frac{j - \tau}{j} \right) \\ &= \frac{e}{(e+1) \cdot k} \cdot \left(1 + \tau \sum_{j=\tau+1}^k \frac{1}{j} \right) \geq \frac{1}{1 + o(1)} \cdot \frac{1}{e+1}. \end{aligned}$$

For $1 < i < \tau$ the corresponding inequality is weaker than the latter inequality. For $i \geq \tau$, we have

$$y_i + \sum_{j=i+1}^k x_j + y_j = y_k + \sum_{j=i+1}^k x_j.$$

Since $y_k = \beta/k$, we therefore have to show that, after scaling x up by a $(1 + o(1))$ -factor, $\sum_{j=i+1}^k x_j$ is at least as large as $(1 - i/k) \cdot \beta$. This is clear for $i = k$. For $\tau \leq i \leq k-1$,

$$\begin{aligned} \sum_{j=i+1}^k x_j &= \frac{e}{(e+1) \cdot k} \cdot \sum_{j=i+1}^k \left(1 - \sum_{\ell=j}^{k-1} \frac{1}{\ell} \right) \\ &= \frac{e}{(e+1) \cdot k} \cdot \left((k - i) - \sum_{j=i+1}^k \sum_{\ell=j}^{k-1} \frac{1}{\ell} \right) \end{aligned}$$

$$\begin{aligned}
&= \frac{e}{(e+1) \cdot k} \cdot \left((k-i) - \sum_{j=i+1}^{k-1} \frac{j-i}{j} \right) \\
&= \frac{e}{(e+1) \cdot k} \cdot \left(1 + i \sum_{j=i+1}^{k-1} \frac{1}{j} \right) \geq \frac{1}{1+o(1)} \cdot \left(1 - \frac{i}{k} \right) \cdot \frac{1}{e+1}.
\end{aligned}$$

Similar to the previous calculations, the objective-function value is

$$\begin{aligned}
&(1+o(1)) \cdot \frac{e}{(e+1) \cdot k} \cdot \sum_{j=\tau}^k \left(1 - \sum_{\ell=j}^{k-1} \frac{1}{\ell} \right) \\
&= (1+o(1)) \cdot \frac{e}{(e+1) \cdot k} \cdot \left(1 + \tau \sum_{j=\tau+1}^k \frac{1}{j} \right) \\
&\leq (1+o(1)) \cdot \frac{e}{(e+1) \cdot k} \cdot \tau \\
&\leq (1+o(1)) \cdot \frac{1}{e+1},
\end{aligned}$$

as claimed. □

5 Conclusion

In this paper, we have established that the 1-2-knapsack secretary problem is no harder than the classic secretary problem in a competitive-ratio sense. While we previously noticed that our technique cannot directly be extended to the general setting, we believe that our work is a first non-trivial step within the larger research plan of settling the achievable competitive ratio for general knapsack secretary.

It seems plausible that our result extends to the setting of arbitrary knapsack size B and item sizes 1 or 2. One approach may be combining our techniques with simple $1/e$ -competitive algorithms for k -secretary [4]. More general variants seem to require handling packings of items of various sizes. A variant that avoids considering such potentially complicated configurations and may still yield an impossibility of larger than $1/e$ is 1- B -knapsack.

References

- [1] S. Albers, A. Khan, and L. Ladewig. Best fit bin packing with random order revisited. *Algorithmica*, 83(9):2833–2858, 2021.
- [2] S. Albers, A. Khan, and L. Ladewig. Improved online algorithms for knapsack and GAP in the random order model. *Algorithmica*, 83(6):1750–1785, 2021.
- [3] S. Albers and L. Ladewig. New results for the k -secretary problem. *Theor. Comput. Sci.*, 863:102–119, 2021.
- [4] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg. A knapsack secretary problem with applications. In *Conference on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 16–28, 2007.
- [5] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg. Matroid secretary problems. *J. ACM*, 65(6):35:1–35:26, 2018.
- [6] H. Böckenhauer, D. Komm, R. Královic, and P. Rossmanith. The online knapsack problem: Advice and randomization. *Theor. Comput. Sci.*, 527:61–72, 2014.
- [7] J. Boyar, L. M. Favrholt, and K. S. Larsen. Online unit profit knapsack with untrusted predictions. In *Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 20:1–20:17, 2022.
- [8] N. Buchbinder, K. Jain, and M. Singh. Secretary problems via linear programming. *Math. Oper. Res.*, 39(1):190–206, 2014.
- [9] T. H. Chan, F. Chen, and S. H. Jiang. Revealing optimal thresholds for generalized secretary problem via continuous LP: impacts on online k -item auction and bipartite k -matching with random arrival order. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1169–1188, 2015.
- [10] J. R. Correa, P. Dütting, F. A. Fischer, and K. Schewior. Prophet inequalities for independent and identically distributed random variables from an unknown distribution. *Math. Oper. Res.*, 47(2):1287–1309, 2022.
- [11] J. R. Correa, P. Dütting, F. A. Fischer, K. Schewior, and B. Ziliotto. Streaming algorithms for online selection problems. In *Innovations in Theoretical Computer Science (ITCS)*, pages 86:1–86:1, 2021.
- [12] M. Dinitz. Recent advances on the matroid secretary problem. *SIGACT News*, 44(2):126–142, 2013.
- [13] E. Dynkin. The optimum choice of the instant for stopping a Markov process. *Soviet Math. Dokl.*, 4, 1963.
- [14] T. Ezra, M. Feldman, N. Gravin, and Z. G. Tang. General graphs are easier than bipartite graphs: Tight bounds for secretary matching. In *ACM Conference on Economics and Computation (EC)*, pages 1148–1177, 2022.
- [15] M. Feldman, O. Svensson, and R. Zenklusen. A simple $O(\log \log(\text{rank}))$ -competitive algorithm for the matroid secretary problem. *Math. Oper. Res.*, 43(2):638–650, 2018.
- [16] T. S. Ferguson. Who solved the secretary problem? *Statistical Science*, 4(3):282–289, 1989.
- [17] J. Giliberti and A. Karrenbauer. Improved online algorithm for fractional knapsack in the random order model. In *Workshop on Approximation and Online Algorithms (WAOA)*, pages 188–205, 2021.
- [18] A. Gupta and S. Singla. Random-order models. In T. Roughgarden, editor, *Beyond the Worst-Case Analysis of Algorithms*, page 234–258. Cambridge University Press, 2021.
- [19] M. Hoefer and B. Kodric. Combinatorial secretary problems with ordinal information. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 133:1–133:14, 2017.
- [20] C. Kenyon. Best-fit bin-packing with random order. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 359–364, 1996.
- [21] T. Kesselheim and M. Molinaro. Knapsack secretary with bursty adversary. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 72:1–72:15, 2020.
- [22] T. Kesselheim, K. Radke, A. Tönnis, and B. Vöcking. An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions. In *European Symposium on Algorithms (ESA)*, pages 589–600, 2013.

- [23] T. Kesselheim, K. Radke, A. Tönnis, and B. Vöcking. Primal beats dual on online packing lps in the random-order model. *SIAM J. Comput.*, 47(5):1939–1964, 2018.
- [24] R. D. Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 630–631, 2005.
- [25] O. Lachish. $O(\log \log \text{rank})$ competitive ratio for the matroid secretary problem. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 326–335, 2014.
- [26] D. Lindley. Dynamic programming and decision theory. *Appl. Statist.*, 10, 1961.
- [27] A. Marchetti-Spaccamela and C. Vercellis. Stochastic on-line knapsack problems. *Math. Program.*, 68:73–104, 1995.
- [28] D. Naori and D. Raz. Online multidimensional packing problems in the random-order model. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 10:1–10:15, 2019.
- [29] J. A. Soto, A. Turkieltaub, and V. Verdugo. Strong algorithms for the ordinal matroid secretary problem. *Math. Oper. Res.*, 46(2):642–673, 2021.
- [30] Y. Zhou, D. Chakrabarty, and R. M. Lukose. Budget constrained bidding in keyword auctions and online knapsack problems. In *International Workshop on Internet and Network Economics (WINE)*, pages 566–576, 2008.

A Proof of Propostion 1

Proof of Propostion 1. In the first case, the optimum consists of a single element, thus, $v(\text{OPT}) = v_1$. Since ALG chooses this element with probability $P_1 \geq p_1$, we have

$$\mathbb{E}[v(\text{ALG})] \geq p_1 \cdot v_1 = p_1 \cdot v(\text{OPT}) \geq (0.35317 \pm o(1)) \cdot v(\text{OPT}), \quad (10)$$

where for the last inequality we used Lemma 1 with $c = 0.26888$.

Now, assume that the optimal packing contains two elements x and y , where we assume $x < y$ w.l.o.g.. Hence, $v(\text{OPT}) = v_x + v_y$. Note that x and y must be the most profitable items among the set of small items, i.e., $r_s(x) = 1$ and $r_s(y) = 2$. Next, we bound the expected profit of the packing. Since $v_i \geq v_x$ for $1 \leq i \leq x$ and $v_i \geq v_y$ for $x+1 \leq i \leq y$, we obtain

$$\mathbb{E}[v(\text{ALG})] \geq \sum_{i=1}^y (P_i \cdot v_i) = v_x \cdot \sum_{i=1}^x P_i + v_y \cdot \sum_{i=x+1}^y P_i.$$

Define $\lambda_x := \sum_{i=1}^x P_i$ and $\lambda_y := \sum_{i=x+1}^y P_i$. If $\lambda_x < \lambda_y$, then $\lambda_y > \lambda_x \geq p_1$ and thus $\mathbb{E}[v(\text{ALG})] \geq v_x \cdot p_1 + v_y \cdot p_1 = p_1 \cdot v(\text{OPT})$, which gives the same bound as in (10). Therefore, we assume $\lambda_x \geq \lambda_y$ in the following. Since $v_x \geq v_y$, applying Chebyshev's sum inequality gives

$$\mathbb{E}[v(\text{ALG})] \geq \left(\frac{v_x + v_y}{2} \right) (\lambda_x + \lambda_y) = \frac{v(\text{OPT})}{2} \cdot \sum_{i=1}^y P_i = \frac{v(\text{OPT})}{2} \cdot \left(2p_y + \sum_{i=1}^y p_i \right),$$

where the last step results from Corollary 1 with $r_s(x) = 1$ and $r_s(y) = 2$. Let $\theta_y = (1/2) \cdot \sum_{i=1}^y p_i + p_y$. By calculating p_i for $1 \leq i \leq 7$ and $c = 0.26888$ using Lemma 1, we obtain the following upper bounds up to additive $o(1)$ terms:

	$y = 2$	$y = 3$	$y = 4$	$y = 5$	$y = 6$	$y = 7$
θ_y	0.4115	0.3820	0.3718	0.3678	0.3662	0.3656

Thus, for $y \in \{2, \dots, 7\}$ we have $\theta_y \geq \theta_7$ for large-enough n . For $y \in \{8, \dots, n\}$, we get $\theta_y = (1/2) \cdot \sum_{i=1}^y p_i + p_y \geq (1/2) \cdot \sum_{i=1}^7 p_i = \theta_7 - p_7$. Hence, for any $y \geq 2$ it holds that $\theta_y \geq \theta_7 - p_7 \geq 0.35317 \pm o(1)$. Overall, in the second case it holds that

$$\mathbb{E}[v(\text{ALG})] \geq \frac{v(\text{OPT})}{2} \cdot \left(2p_y + \sum_{i=1}^y p_i \right) \geq (\theta_7 - p_7) v(\text{OPT}) \geq (0.35317 \pm o(1)) \cdot v(\text{OPT}).$$

This completes the proof. □