Improved Online Algorithms for Knapsack and GAP in the Random Order Model

Susanne Albers
Technical University of Munich, Germany
albers@in.tum.de

Arindam Khan
Indian Institute of Science, Bangalore, India[1]
arindamkhan@iisc.ac.in

Leon Ladewig
Technical University of Munich, Germany
ladewig@in.tum.de

Abstract

The *knapsack problem* is one of the classical problems in combinatorial optimization: Given a set of items, each specified by its size and profit, the goal is to find a maximum profit packing into a knapsack of bounded capacity. In the online setting, items are revealed one by one and the decision, if the current item is packed or discarded forever, must be done immediately and irrevocably upon arrival. We study the online variant in the random order model where the input sequence is a uniform random permutation of the item set.

We develop a randomized (1/6.65)-competitive algorithm for this problem, outperforming the current best algorithm of competitive ratio 1/8.06 [Kesselheim et al. SIAM J. Comp. 47(5)]. Our algorithm is based on two new insights: We introduce a novel algorithmic approach that employs two given algorithms, optimized for restricted item classes, sequentially on the input sequence. In addition, we study and exploit the relationship of the knapsack problem to the 2-secretary problem.

The *generalized assignment problem* (GAP) includes, besides the knapsack problem, several important problems related to scheduling and matching. We show that in the same online setting, applying the proposed sequential approach yields a (1/6.99)-competitive randomized algorithm for GAP. Again, our proposed algorithm outperforms the current best result of competitive ratio 1/8.06 [Kesselheim et al. SIAM J. Comp. 47(5)].

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Online algorithms, knapsack problem, random order model

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2019.22

Category APPROX

Funding Work supported by the European Research Council, Grant Agreement No. 691672.

1 Introduction

Many real-world problems can be considered resource allocation problems. For example, consider the loading of cargo planes with (potential) goods of different weights. Each item raises a certain profit for the airline if it is transported; however, not all goods can be loaded due to airplane weight restrictions. Clearly, the dispatcher seeks for a maximum profit packing fulfilling the capacity constraint. This example from [24] illustrates the *knapsack problem*: Given a set of n items, specified by a size and a profit value, and a resource (called knapsack) of fixed capacity, the goal is to find a subset of items (called packing) with maximum total

[1] A part of this work was done when the author was at Technical University of Munich.

profit and whose total size does not exceed the capacity. Besides being a fundamental and extensively studied problem in combinatorial optimization, knapsack problems arise in many and various practical settings. We refer the readers to textbooks [24, 35] and to the surveys of previous work in [14, 19] for further references.

In the *generalized assignment problem* (GAP) [35], resources of different capacities are given, and the size and the profit of an item depend on the resource to which it is assigned. The GAP includes many prominent problems, such as the (multiple) knapsack problem [13], weighted bipartite matching [28], AdWords [36], and the display ads problem [17]. Further applications of GAP are outlined in the survey articles [11, 41].

We study online variants of the knapsack and GAP problems. Here, n items are presented sequentially, and the decision for each item must be made immediately upon arrival. In fact, many real-world optimization problems occur as online problems, as often decisions must be made under uncertain conditions. For example, consider the introducing logistics example, if the airline needs to answer customer requests immediately without knowing future requests. The online knapsack problem has been studied in particular in the context of online auctions [9, 45].

Typically, the performance measure for online algorithms is the *competitive ratio*, which is defined as the ratio between the values of the algorithmic solution and an optimal offline solution for a worst-case input. It can be shown that, even for the knapsack problem, the general online setting admits no algorithms with bounded competitive ratio [34, 45]. However, most hardness results are based on a worst-case input presented in adversarial order. In the *random order model*, the performance of an algorithm is evaluated for a worst-case input, but the adversary has no control over the input order; the input sequence is drawn uniformly at random among all permutations. This model is known from the secretary problem [15, 31] and its generalizations [7, 12, 18]; it has been successfully applied to other online problems, for example, scheduling and packing [1, 16, 20, 25, 27, 39], graph problems [8, 26, 33], facility location [37], budgeted allocation [38], and submodular welfare maximization [30].

1.1 Related Work

Online knapsack problem. The problem was first studied by Marchetti-Spaccamela and Vercellis [34], who showed that no deterministic online algorithm for this problem can obtain a constant competitive ratio. Moreover, Chakrabarty et al. [45] demonstrated that this fact cannot be overcome by randomization.

Given such hardness results, several relaxations have been introduced and investigated. Most relevant to our work are results in the random order model. Introduced as the *secretary knapsack problem* [6], Babaioff et al. developed a randomized algorithm of competitive ratio $1/(10e) < 1/27$. Kesselheim et al. [27] achieved a significant improvement by developing a $(1/8.06)$-competitive randomized algorithm for the generalized assignment problem. Finally, Vaze [43] showed that there exists a deterministic algorithm of competitive ratio $1/(2e) < 1/5.44$, assuming that the maximum profit of a single item is small compared to the profit of the optimal solution.

Apart from the random order model, different further relaxations have been considered. Marchetti-Spaccamela and Vercellis [34] studied a stochastic model wherein item sizes and profits are drawn from a fixed distribution. Lueker [32] obtained improved bounds in this model. Chakrabarty et al. [45] studied the problem when the density (profit-size ratio) of each item is in a fixed range $[L, U]$. Under the further assumption that item sizes are small compared to the knapsack capacity, Chakrabarty et al. proposed an algorithm of competitive ratio $\ln(U/L) + 1$ and provided a lower bound of $\ln(U/L)$. Another branch of

research considers removable models, where the algorithm can remove previously packed items. Removing such items can incur no cost [22,23] or a cancellation cost (*buyback model*, [4,5,21]). Recently, Vaze [44] considered the problem under a (weaker) expected capacity constraint. This variant admits a competitive ratio of $1/4e$.

Online GAP. Since all hardness results for online knapsack also hold for online GAP, research focuses on stochastic variants or modified online settings. Currently, the only result for the random order model is the previously mentioned (1/8.06)-competitive randomized algorithm proposed by Kesselheim et al. [27]. To the best of our knowledge, the earliest paper considering online GAP is due to Feldman et al. [17]. They obtained an algorithm of competitive ratio tending to $1 - 1/e$ in the *free disposal model*. In this model, the total size of items assigned to a resource might exceed its capacity; in addition, no item consumes more than a small fraction of any resource. A stochastic variant of online GAP was studied by Alaei et al. [2]. Here, the size of an item is drawn from an individual distribution that is revealed upon arrival of the item, together with its profit. However, the algorithm learns the actual item size only after the assignment. If no item consumes more than a $(1/k)$-fraction of any resource, the algorithm proposed by Alaei et al. has competitive ratio $1 - 1/\sqrt{k}$.

Online packing LPs. In contrast to GAP, general packing LPs describe problems where requests can consume more than one resource. The study of online packing LPs was initiated by Buchbinder and Naor [10] in the adversarial model. In several papers [1, 16, 27, 39] it has been shown that the random order model admits $(1 - \varepsilon)$-competitive algorithms assuming large capacity ratios, i.e., when the capacity of any resource is large compared to the maximum demand for it. Most recently, Kesselheim et al. [27] showed that there is a $(1 - \varepsilon)$-competitive algorithm if $B = \Omega((\log d)/\varepsilon^2)$, where B is the capacity ratio and d is the column sparsity (the maximum number of resources occurring in a single column).

1.2 Our Contributions

As outlined above, for online knapsack and GAP in the adversarial input model, nearly all previous works attain constant competitive ratios at the cost of either (a) imposing structural constraints on the input or (b) significantly relaxing the original online model. Therefore, we study both problems in the random order model, which is less pessimistic than the adversarial model but still considers worst-case instances without further constraints on the item properties. For the knapsack problem, our main result is the following.

▶ **Theorem 1.1.** *There exists a* (1/6.65)*-competitive randomized algorithm for the online knapsack problem in the random order model assuming* $n \to \infty$.

One challenge in the design of knapsack algorithms is that the optimal packing can have, on a high level, at least two different structures. Either there are few large items, constituting the majority of the packing's profit, or there are many small such items. Previous work [6,27] is based on splitting the input according to item sizes and then employing algorithms tailored for these restricted instances. However, the algorithms from [6,27] choose a single item type via an initial random choice, and then pack items of that type exclusively. In contrast, our approach considers different item types in distinct time intervals, rather than discarding items of a specific type in advance. More precisely, we develop algorithms $\mathcal{A}_L$ and $\mathcal{A}_S$ which are combined in a novel *sequential approach*: While large items appearing in early rounds are packed using $\mathcal{A}_L$, algorithm $\mathcal{A}_S$ is applied to pack small items revealed in later rounds. We think that this approach may be helpful for other problems in similar online settings as well.

The proposed algorithm $\mathcal{A}_L$ deals with the knapsack problem where all items consume more than 1/3 of the capacity (we call this problem 2-KS). The 2-KS problem is closely related to the k-secretary problem [29] for $k = 2$. We also develop a general framework that allows to employ any algorithm for the 2-secretary problem to obtain an algorithm for 2-KS. As a side product, we obtain a simple (1/3.08)-competitive deterministic algorithm for 2-KS in the random order model. For items whose size is at most 1/3 of the resource capacity, we give a simple and efficient algorithm $\mathcal{A}_S$. Here, a challenging constraint is that $\mathcal{A}_L$ and $\mathcal{A}_S$ share the same resource, so we need to argue carefully that the decisions of $\mathcal{A}_S$ are feasible, given the packing of $\mathcal{A}_L$ from previous rounds.

Finally, we show that the proposed sequential approach also improves the current best result for GAP [27] from competitive ratio 1/8.06 to 1/6.99.

▶ **Theorem 1.2.** *There exists a* (1/6.99)*-competitive randomized algorithm for the online generalized assignment problem in the random order model assuming* $n \to \infty$.

For this problem we use the algorithmic building blocks $\mathcal{A}_L$, $\mathcal{A}_S$ developed in [26,27]. However, we need to verify that $\mathcal{A}_L$, an algorithm for edge-weighted bipartite matching [26], satisfies the desired properties for the sequential approach. We point out that the assignments of our algorithm differ structurally from the assignments of the algorithm proposed in [27]. In the assignments of the latter algorithm, all items are either large or small compared to the capacity of the assigned resource. In our approach, both situations can occur, because resources are managed independently.

Roadmap. We focus on the result on the knapsack problem (Theorem 1.1) in the first chapters of this paper. For this purpose, we provide elementary definitions in Section 2. Our main technical contribution is formally introduced in Section 3: Here, we describe an algorithmic framework performing two algorithms $\mathcal{A}_L$, $\mathcal{A}_S$ sequentially. In Sections 4 and 5, we design and analyze the algorithms $\mathcal{A}_L$ and $\mathcal{A}_S$ for the knapsack problem. Finally, in Section 6 we describe how the sequential approach can be applied to GAP. Due to space constraints, some proofs are deferred to Appendix A (knapsack) and to Appendix B (GAP).

2 Preliminaries

Let $[n] := \{1, \dots, n\}$. Further, let $\mathbb{Q}_{\geq 0}$ and $\mathbb{Q}_{>0}$ denote the set of non-negative and positive rational numbers, respectively.

Knapsack problem. We are given a set of items $I = [n]$, each item $i \in I$ has *size* $s_i \in \mathbb{Q}_{>0}$ and a *profit* (*value*) $v_i \in \mathbb{Q}_{\geq 0}$. The goal is to find a maximum profit packing into a knapsack of size $W \in \mathbb{Q}_{>0}$, i.e., a subset $M \subseteq I$ such that $\sum_{i \in M} s_i \leq W$ and $\sum_{i \in M} v_i$ is maximized. W.l.o.g. we can assume $s_i \leq W$ for all $i \in I$. In the online variant of the problem, in each *round* $\ell \in [n]$ a single item i is revealed together with its size and profit. The online algorithm must decide immediately and irrevocably whether to pack i. We call an item *visible in round* ℓ if it arrived in round ℓ or earlier.

Random order performance. We analyze the performance of algorithms in the *random order model*. Given a worst case input $\mathcal{I}$, the order in which $\mathcal{I}$ is presented is drawn uniformly at random from the set of all permutations. For an algorithm $\mathcal{A}$, its *competitive ratio* is defined as $\mathbf{E}[\mathcal{A}(\mathcal{I})] / \mathrm{OPT}(\mathcal{I})$, where $\mathcal{A}(\mathcal{I})$ and $\mathrm{OPT}(\mathcal{I})$ denote the profits of the solutions of $\mathcal{A}$ and an optimal offline algorithm, respectively. Here, the expectation is taken over

Algorithm 1 Sequential approach.

Input : Random permutation π of n items in I, a knapsack of capacity W, parameters $c, d \in (0,1)$ with $c < d$, algorithms $\mathcal{A}_L$, $\mathcal{A}_S$.
Output : A feasible (integral) knapsack packing.
Let ℓ be the current round.
if $\ell \leq cn$ **then**
  Sampling phase – discard all items;
if $cn + 1 \leq \ell \leq dn$ **then**
  Pack $\pi(\ell)$ iff $\mathcal{A}_L$ packs $\pi_L(\ell)$;
if $dn + 1 \leq \ell \leq n$ **then**
  Pack $\pi(\ell)$ iff $\mathcal{A}_S$ packs $\pi_S(\ell)$ and the remaining capacity is sufficiently large.

all permutations and random choices of the algorithm. As above, we slightly overload the notation and also use $\mathcal{A}$ as a random variable for the profit of the solution returned by an algorithm $\mathcal{A}$.

We classify items as large or small, depending on their size compared to W and a parameter $\delta \in (0,1)$ to be determined later.

▶ **Definition 2.1.** *We say an item i is δ-large if $s_i > \delta W$ and δ-small if $s_i \leq \delta W$. Whenever δ is clear from the context, we say an item is large or small for short. Based on the given item set I, we define two modified item sets I_L and I_S, which are obtained as follows:*

- *I_L: Replace each small item by a large item of profit 0*
- *I_S: Replace each large item by a small item of profit 0.*

Therefore, I_L only contains large items and I_S only contains small items. We can assume that no algorithm packs a zero-profit item, thus any algorithmic packing of I_L or I_S can be turned into a packing of I having the same profit. Let OPT, OPT_L, and OPT_S be the total profits of optimal packings for I, I_L, and I_S, respectively. A useful upper bound for OPT is

$$\mathrm{OPT} \leq \mathrm{OPT}_L + \mathrm{OPT}_S\,. \tag{1}$$

3 Sequential Approach

A common approach in the design of algorithms for secretary problems is to set two phases: a *sampling phase*, where all items are rejected, followed by a *decision phase*, where some items are accepted according to a decision rule. Typically, this rule is based on the information gathered in the sampling phase. We take this concept a step further: The key idea of our sequential approach is to use a part of the sampling phase of one algorithm as decision phase of another algorithm, which itself can have a sampling phase. This way, two algorithms are performed in a sequential way, which makes better use of the entire instance. We combine this idea with using different strategies for small and large items.

Formally, let $\mathcal{A}_L$ and $\mathcal{A}_S$ be two online knapsack algorithms and I_L and I_S be the item sets constructed according to Definition 2.1. Further, let $0 < c < d < 1$ be two parameters to be specified later. Our proposed algorithm samples the first cn rounds; during this time no item is packed. From round $cn + 1$ to dn, the algorithm considers large items exclusively. In this interval we follow the decisions of $\mathcal{A}_L$. After round dn, the algorithm processes only small items and follows the decisions of $\mathcal{A}_S$. However, it might be the case that an item accepted by $\mathcal{A}_S$ cannot be packed because the knapsack capacity is exhausted due to the packing of $\mathcal{A}_L$ in earlier rounds. Note that all rounds $1, \ldots, dn$ can be considered as the

Algorithm 2 Algorithm $\mathcal{A}_L$ for large items.

Input : Random permutation of n $(1/3)$-large items, a knapsack of capacity W, parameters $c, d \in (0,1)$ with $c < d$.
Output : A feasible (integral) packing of the knapsack.
Let ℓ be the current round.
if $\ell \leq cn$ **then**
  Sampling phase – discard all items.
Let v^* be the maximum profit seen up to round cn.
if $cn + 1 \leq \ell \leq dn$ **then**
  Pack the first two items of profit higher than v^*, if feasible.
if $\ell > dn$ **then**
  Discard all items.

sampling phase for $\mathcal{A}_S$. A formal description is given in Algorithm 1. Here, for a given input sequence π of I, let π_L and π_S denote the corresponding sequences from I_L and I_S, respectively. Note that π is revealed sequentially and π_L, π_S can be constructed online. For any input sequence π, let $\pi(\ell)$ denote the item at position $\ell \in [n]$.

In the final algorithm we set the threshold for small items to $\delta = 1/3$ and use Algorithm 1 with parameters $c = 0.42291$ and $d = 0.64570$. Under the assumption $n \to \infty$ we can assume $cn, dn \in \mathbb{N}$. We next give a high-level description of the proof of Theorem 1.1.

Proof of Theorem 1.1. Let $\mathcal{A}$ be Algorithm 1 and $\mathcal{A}_L$, $\mathcal{A}_S$ be the algorithms developed in Sections 4 and 5. In the next sections we prove the following results (see Lemmas 4.7 and 5.5): The expected profit from $\mathcal{A}_L$ in rounds $cn+1, \dots, dn$ is at least $\frac{1}{6.65}\,\mathrm{OPT}_L$, and the expected profit from $\mathcal{A}_S$ in rounds $dn+1, \dots, n$ is at least $\frac{1}{6.65}\,\mathrm{OPT}_S$. Together with inequality (1), we obtain

$$\mathbf{E}[\mathcal{A}] \geq \mathbf{E}[\mathcal{A}_L] + \mathbf{E}[\mathcal{A}_S] \geq \frac{1}{6.65}\,\mathrm{OPT}_L + \frac{1}{6.65}\,\mathrm{OPT}_S \geq \frac{1}{6.65}\,\mathrm{OPT}\,. \qquad \blacktriangleleft$$

The order in which $\mathcal{A}_L$ and $\mathcal{A}_S$ are arranged in Algorithm 1 follows from two observations. Algorithm $\mathcal{A}_S$ is powerful if it samples roughly $(2/3)n$ rounds; a part of this long sampling phase can be used as the decision phase of $\mathcal{A}_L$, for which a shorter sampling phase is sufficient. Moreover, the first algorithm should either pack high-profit items, or should leave the knapsack empty for the following algorithm with high probability. The algorithm $\mathcal{A}_L$ we propose in Section 4 has this property (see Lemma 4.8). In contrast, if $\mathcal{A}_S$ would precede $\mathcal{A}_L$, the knapsack would be empty at the beginning of $\mathcal{A}_L$ with very small probability, in which case we would not benefit from $\mathcal{A}_L$.

Finally, note that better algorithms and parameterizations for the respective sub-problems exist (see Lemma 4.6 and [27]). However, for the overall performance we need algorithms $\mathcal{A}_L$ and $\mathcal{A}_S$ that perform well evaluated in the sequential framework.

4 Large Items

The approach presented in this section is based on the connection between the online knapsack problem under random arrival order and the k-secretary problem [29]. In the latter problem, the algorithm can accept up to k items and the goal is to maximize the sum of their profits. The k-secretary problem generalizes the classical secretary problem [15, 31] and is itself a special case of the online knapsack problem under random arrival order (if all knapsack items have size W/k).

Table 1 Definition of packing types A-M. We use set notation $\{i, j\}$ if i and j can be packed in any order, and tuple notation (i, j) if the packing order must be as given.

type	content	constraint on j	probability p_X
A	$\{1, 2\}$	-	$p_{12} + p_{21}$
B	$\{1, 3\}$	-	$p_{13} + p_{31}$
C	$\{2, 3\}$	-	$p_{23} + p_{32}$
D	$(1, j)$	-	p_1
E	$(2, j)$	-	p_2
F	$(3, j)$	-	p_3
G	$(4, j)$	-	p_4
H	$(1, j)$	$j \neq 2$	$p_1 - p_{12}$
I	$(1, j)$	$j \neq 3$	$p_1 - p_{13}$
J	$(2, j)$	$j \neq 1$	$p_2 - p_{21}$
K	$(2, j)$	$j \neq 3$	$p_2 - p_{23}$
L	$(3, j)$	$j \neq 1$	$p_3 - p_{31}$
M	$(3, j)$	$j \neq 2$	$p_3 - p_{32}$

In our setting, each large item consumes more than $\delta = 1/3$ of the knapsack capacity. We call this problem 2-KS, since at most two items can be packed completely. Therefore, any 2-secretary algorithm can be employed to identify high-profit items and pack them if feasible. Although this idea applies to any δ and corresponding k, the approach seems stronger for small k: Intuitively, the characteristics of k-KS and k-secretary deviate with growing k, while 1-KS is exactly 1-secretary. Furthermore, the k-secretary problem is for $k = 2$ rather well studied [3, 12], while the exact optimal competitive ratios for $k \geq 3$ are still unknown.

In the following, let $\mathcal{A}_L$ be Algorithm 2. This is an adaptation of the algorithm SINGLE-REF developed for the k-secretary problem in [3]. As discussed above, 2-secretary and 2-KS are similar, but different problems. Therefore, in our setting it is not possible to apply the existing analysis from [3] or from any other k-secretary algorithm directly.

Assumption. For this section we assume that all profits are distinct. This is without loss of generality, as ties can be broken by adjusting the profits slightly, using the items' identifiers. Further, we assume $v_1 > v_2 > \ldots > v_n$ and say that i is the *rank* of item i.

4.1 Packing Types

As outlined above, in contrast to the 2-secretary problem, not all combinations of two knapsack items can be packed completely. Therefore, we analyze the probability that $\mathcal{A}_L$ selects a feasible set of items whose profit can be bounded from below. We restrict our analysis to packings where an item $i \in \{1, 2, 3, 4\}$ is packed as the first item and group such packings into several packing types A-M defined in the following. Although covering more packings might lead to further insights into the problem and to a stronger result, we expect the improvement to be marginal.

Let p_X be the probability that $\mathcal{A}_L$ returns a packing of type $X \in \{\mathrm{A}, \ldots, \mathrm{M}\}$. In addition, let p_i for $i \in [n]$ be the probability that $\mathcal{A}_L$ packs i as the first item. Finally, let p_{ij} for $i, j \in [n]$ be the probability that $\mathcal{A}_L$ packs i as the first item and j as the second item.

In a packing of type A, the items 1 and 2 are packed in any order. Therefore, $p_{\mathrm{A}} = p_{12} + p_{21}$. The types B and C are defined analogously using the items $\{1, 3\}$ and $\{2, 3\}$, respectively. In a packing of type D, the item 1 is accepted as the first item, together with no or any second

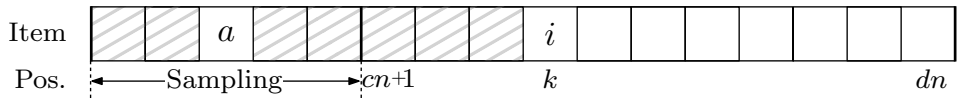

Figure 1 Input sequence considered in Lemma 4.2. The gray dashed slots represent items of rank greater than a.

item j. This happens with probability $p_{\mathrm{D}} = p_1$. Accordingly, we define types E,F, and G using the items 2,3, and 4, respectively. Finally, for each item $i \in \{1,2,3\}$, we introduce two further packing types. For $i = 1$, types H and I characterize packings where the first accepted item is 1, the second accepted item j is not 2 (type H) and not 3 (type I), respectively. Therefore, we get $p_{\mathrm{H}} = p_1 - p_{12}$ and $p_{\mathrm{I}} = p_1 - p_{13}$. Packing types J-K and L-M describe analogous packings for $i = 2$ and $i = 3$, respectively. Table 1 shows all packing types A-M and their probabilities expressed by p_i and p_{ij}.

The packing types defined above allow to describe all packings where a specific item $i \in \{1,2,3,4\}$ is packed as the first item, without covering the same packing multiple times. For example, packing types A and D (with $j = 2$) both include the packing $(1,2)$; however, we can consider the disjoint packing types A and H.

4.2 Acceptance Probabilities of Algorithm 2

In the following we compute the probabilities p_i and p_{ij} from Table 1 as functions of c and d. Throughout the following proofs, we denote the position of an item i in a given permutation with $\mathrm{pos}(i) \in [n]$. Further, let a be the maximum profit item from sampling.

We think of the random permutation as being sequentially constructed. The fact given below follows from the hypergeometric distribution and becomes helpful in the proofs of Lemmas 4.2 and 4.3.

▶ **Fact 4.1.** *Suppose there are N balls in an urn from which M are blue and $N - M$ red. The probability of drawing K blue balls without replacement in a sequence of length K is $h(N,M,K) := \binom{M}{K}/\binom{N}{K}$.*

In the first lemma, we provide the probabilities p_i for $i \in [4]$ assuming $n \to \infty$.

▶ **Lemma 4.2.** *Assuming $n \to \infty$, it holds that*

$$p_i = \begin{cases} c \ln \frac{d}{c} & i = 1 \\ c\left(\ln \frac{d}{c} - d + c\right) & i = 2 \\ c\left(\ln \frac{d}{c} - 2(d-c) + \frac{1}{2}(d^2 - c^2)\right) & i = 3 \\ c\left(\ln \frac{d}{c} - 3(d-c) + \frac{3}{2}(d^2 - c^2) - \frac{1}{3}(d^3 - c^3)\right) & i = 4\,. \end{cases}$$

Proof. We construct the random permutation by drawing the positions for items sequentially, starting with the items i and a. For any position $k \geq cn + 1$, the permutation fulfills $\mathrm{pos}(i) = k$ and $\mathrm{pos}(a) \leq cn$ with probability $\frac{1}{n}\frac{cn}{n-1} = \frac{c}{n-1}$. Next, we draw the remaining $k - 2$ items for the slots up to position k. Since i is packed as the first item, all previous items (except for a) must have rank greater than a (see Figure 1). As these items are drawn from the remaining $n - 2$ items (of which $n - a$ have rank greater than a), the probability for this step is $h(n-2, n-a, k-2)$ according to Fact 4.1. Using the law of total probability for $k \in \{cn+1, \ldots, dn\}$ and $a \in \{i+1, \ldots, n\}$ we obtain

$$p_i = \frac{c}{n-1} \sum_{k=cn+1}^{dn} \sum_{a=i+1}^{n} h(n-2, n-a, k-2) = \frac{c}{n-1} \sum_{k=cn+1}^{dn} \frac{1}{\binom{n-2}{k-2}} \sum_{a=i+1}^{n} \binom{n-a}{k-2}.$$

We can simplify this term further by observing

$$\sum_{a=i+1}^{n} \binom{n-a}{k-2} = \sum_{a=0}^{n-i-1} \binom{a}{k-2} = \binom{n-i}{k-1}.$$

Therefore, $p_i = \frac{c}{n-1} \sum_{k=cn+1}^{dn} \binom{n-i}{k-1} / \binom{n-2}{k-2}$.

Asymptotics. It holds that

$$\lim_{n\to\infty} \frac{\binom{n-i}{k-1}}{\binom{n-2}{k-2}} = \lim_{n\to\infty} \frac{(n-i)!}{(n-2)!} \frac{(n-k)!}{(n-i-k+1)!} \frac{1}{k-1} = \frac{(n-k)^{i-1}}{n^{i-2}} \frac{1}{k}.$$

Hence, $\lim_{n\to\infty} p_i = (c/n^{i-1}) \sum_{k=cn+1}^{dn} f(k)$ where $f(k) := (n-k)^{i-1}/k$. Since f is monotonically decreasing in k, we have $\int_{cn+1}^{dn+1} f(k)\,\mathrm{d}k \leq \sum_{k=cn+1}^{dn} f(k) \leq \int_{cn}^{dn} f(k)\,\mathrm{d}k$. Let F be a function such that $\int_a^b f(k)\,\mathrm{d}k = F(b) - F(a)$ for $0 < a < b$. As it holds that $\lim_{n\to\infty} F(dn+1) - F(dn) = \lim_{n\to\infty} F(cn+1) - F(cn) = 0$, the above bounds are asymptotically tight, i.e., $\lim_{n\to\infty} \sum_{k=cn+1}^{dn} f(k) = F(dn) - F(cn)$. Below we give functions F for $i \in [4]$.

i	$f(k)$	$F(k)$	$F(dn) - F(cn)$
1	$\frac{1}{k}$	$\ln k$	$\ln \frac{d}{c}$
2	$\frac{n-k}{k}$	$n \ln k - k$	$n \ln \frac{d}{c} - dn + cn$
3	$\frac{(n-k)^2}{k}$	$n^2 \ln k - 2nk + \frac{k^2}{2}$	$n^2 \ln \frac{d}{c} - 2n(dn - cn) + \frac{d^2n^2 - c^2n^2}{2}$
4	$\frac{(n-k)^3}{k}$	$n^3 \ln k - 3n^2k + \frac{3}{2}nk^2 - \frac{k^3}{3}$	$n^3 \ln \frac{d}{c} - 3n^3(d-c) + \frac{3}{2}n^3(d^2 - c^2) - \frac{1}{3}n^3(d^3 - c^3)$

The claims follow by multiplying the respective terms with c/n^{i-1}. ◀

Next, we analyze the probabilities p_{ij} for $i \neq j$ and $i, j \in [3]$. The next lemma deals with the cases where $j = i + 1$.

▶ **Lemma 4.3.** *For $n \to \infty$ it holds that*

$$p_{12} = c\left(d - c\ln\frac{d}{c} - c\right),$$
$$p_{23} = c\left(d - c\ln\frac{d}{c} - c - \frac{d^2}{2} + cd - \frac{c^2}{2}\right).$$

The proof of Lemma 4.3 is technically similar to the proof of Lemma 4.2 and thus deferred to Appendix A. It remains to analyze the probabilities p_{13}, p_{31}, p_{21}, and p_{32}. Interestingly, they all reduce to the two probabilities considered in Lemma 4.3. The following two lemmas should be intuitively clear from the description of Algorithm 2. For completeness, we give formal proofs in Appendix A.

▶ **Lemma 4.4.** *For any two items i and j it holds that $p_{ij} = p_{ji}$.*

▶ **Lemma 4.5.** *For any three items $i < k < j$ it holds that $p_{ij} = p_{kj}$.*

Therefore, we have $p_{13} = p_{23}$ by Lemma 4.5 and $p_{31} = p_{13}$, $p_{21} = p_{12}$, and $p_{32} = p_{23}$ by Lemma 4.4.

4.3 Analysis

Let T be the set of items in the optimal packing of I_L. This set may contain a single item, may be a two-item subset of $\{1,2,3\}$, or may be a two-item subset containing an item $j \geq 4$. In the following we analyze the performance of Algorithm 2 for each case.

Single-item case. Let case 1 be the case where $T = \{1\}$. In case 1, $\mathbf{E}[\mathcal{A}_L] \geq p_{\mathrm{D}} \operatorname{OPT}_L$.

Two-item cases. In cases 2–4, we consider packings of the form $T = \{i,j\}$ with $1 \leq i < j \leq 3$. We define cases 2, 3, and 4 as $T = \{1,2\}$, $T = \{1,3\}$, and $T = \{2,3\}$, respectively. We want to consider all algorithmic packings whose profit can be bounded in terms of $\operatorname{OPT}_L = v_i + v_j$. For this purpose, for each case 2-4 we build three groups of feasible packing types, according to whether the profit of a packing is OPT_L, at least v_i, or in the interval $(v_i, v_j]$. We ensure that no packing is counted multiple times by (a) choosing appropriate packing types and (b) grouping these packing types in a disjoint way, according to their profit. Let α_w be the probability that the algorithm returns the optimal packing in case $w \in \{2,3,4\}$. It holds that $\alpha_2 = p_{\mathrm{A}}$, $\alpha_3 = p_{\mathrm{B}}$, and $\alpha_4 = p_{\mathrm{C}}$. In addition, let β_w be the probability that an item $k \leq i$ is packed as the first item in case $w \in \{2,3,4\}$. We have $\beta_2 = p_{\mathrm{H}}$, $\beta_3 = p_{\mathrm{I}}$, and $\beta_4 = p_{\mathrm{D}} + p_{\mathrm{K}}$. Finally, let γ_w be the probability that an item k with $i < k \leq j$ is packed as the first item in case $w \in \{2,3,4\}$. It holds that $\gamma_2 = p_{\mathrm{J}}$, $\gamma_3 = p_{\mathrm{E}} + p_{\mathrm{L}}$, and $\gamma_4 = p_{\mathrm{M}}$.

Finally, we define case 5 as $T = \{i,j\}$ with $i \geq 1$, $j \geq 4$, and $i < j$. In this case, note that packings of type D contain an item of value at least v_i, and packings of type E, F, and G contain an item of value at least v_j. Hence, we can slightly abuse the notation and set $\alpha_5 = 0$, $\beta_5 = p_{\mathrm{D}}$, and $\gamma_5 = p_{\mathrm{E}} + p_{\mathrm{F}} + p_{\mathrm{G}}$, such that it holds that

$$\mathbf{E}[\mathcal{A}_L] \geq \alpha_w(v_i + v_j) + \beta_w v_i + \gamma_w v_j \qquad \text{in case } w \in \{2,3,4,5\}.$$

To bound this term against $\operatorname{OPT}_L = v_i + v_j$, consider the following two cases: If $\beta_w \geq \gamma_w$, we obtain from Chebyshev's sum inequality $\beta_w v_i + \gamma_w v_j \geq \frac{1}{2}(\beta_w + \gamma_w)(v_i + v_j)$. If $\beta_w < \gamma_w$, we trivially have $\beta_w v_i + \gamma_w v_j > \beta_w(v_i + v_j)$. Thus, we obtain

$$\mathbf{E}[\mathcal{A}_L] \geq \left(\alpha_w + \min\left\{\frac{\beta_w + \gamma_w}{2}, \beta_w\right\}\right) \operatorname{OPT}_L \qquad \text{in case } w \in \{2,3,4,5\}. \tag{2}$$

The competitive ratio of $\mathcal{A}_L$ is the minimum over all cases 1-5. We obtain the following two lemmas. If the algorithm is allowed to use the entire input sequence ($d = 1$), $\mathcal{A}_L$ has a competitive ratio of $1/3.08$.

▶ **Lemma 4.6.** *With $c = 0.23053$ and $d = 1$, algorithm $\mathcal{A}_L$ satisfies $\mathbf{E}[\mathcal{A}_L] \geq \frac{1}{3.08} \operatorname{OPT}_L$.*

Note that 2-KS includes the secretary problem (case 1); thus, no algorithm for 2-KS can have a better competitive ratio than $1/e < 1/2.71$. In the final algorithm we set $d < 1$ to benefit from $\mathcal{A}_S$. The next lemma has already been used to prove Theorem 1.1 in Section 3.

▶ **Lemma 4.7.** *With $c = 0.42291$ and $d = 0.64570$, algorithm $\mathcal{A}_L$ satisfies $\mathbf{E}[\mathcal{A}_L] \geq \frac{1}{6.65} \operatorname{OPT}_L$.*

Proof of Lemmas 4.6 and 4.7. For the overall competitive ratio, we build the minimum over all cases. According to inequality (2), the competitive ratios for the two-item cases depend on $\beta_w \geq \gamma_w$ or $\beta_w < \gamma_w$. However, for the parameter pairs $(c,d) = (0.23053, 1)$ from

Table 2 Competitive ratios of Algorithm 2 for the parameters from Lemmas 4.6 and 4.7 in different cases. Bold values indicate the minimum over all cases and thus the competitive ratio.

				two-item cases			
	c	d	case 1	case 2	case 3	case 4	case 5
Lemma 4.6	0.23053	1	0.33827	0.34898	0.32705	0.32705	**0.32471**
Lemma 4.7	0.42291	0.64570	0.17897	**0.15039**	0.16033	0.16033	0.16231

Lemma 4.6 and $(c, d) = (0.42291, 0.64570)$ from Lemma 4.7 we have $\beta_w \geq \gamma_w$ for any case $w \in \{2, 3, 4, 5\}$. This follows from a technical lemma provided in Appendix A (Lemma A.1). Hence, inequality (2) simplifies to $\mathbf{E}[\mathcal{A}_L] \geq \left(\alpha_w + \frac{\beta_w + \gamma_w}{2}\right) \mathrm{OPT}_L$ in case $w \in \{2, 3, 4, 5\}$. Using the definitions of p_X from Table 1 and the symmetry property of Lemma 4.4 we get

$$\mathbf{E}[\mathcal{A}_L] / \mathrm{OPT}_L \geq \begin{cases} p_1 & \text{case 1} \\ p_{12} + (p_1 + p_2)/2 & \text{case 2} \\ p_{13} + (p_1 + p_2 + p_3)/2 & \text{case 3} \\ p_{23} + (p_1 + p_2 + p_3)/2 & \text{case 4} \\ (p_1 + p_2 + p_3 + p_4)/2 & \text{case 5} \ . \end{cases} \tag{3}$$

Note that the algorithm attains the same competitive ratio in case 3 and 4, since $p_{13} = p_{23}$. Table 2 shows the competitive ratios for all five cases obtained from Equation (3). For the overall competitive ratio, we have

$$\mathbf{E}[\mathcal{A}_L] \geq \min\left\{p_1, p_{12} + \frac{p_1 + p_2}{2}, p_{23} + \frac{p_1 + p_2 + p_3}{2}, \frac{p_1 + p_2 + p_3 + p_4}{2}\right\} \mathrm{OPT}_L \ .$$

Hence, the competitive ratios are $0.32471 \geq 1/3.08$ and $0.15039 \geq 1/6.65$ for Lemma 4.6 and Lemma 4.7, respectively. ◀

Recall that in Algorithm 1, we can only benefit from $\mathcal{A}_S$ if $\mathcal{A}_L$ has not filled the knapsack completely. Thus, the following property is crucial in the final analysis.

▶ **Lemma 4.8.** *With probability of at least c/d, no item is packed by $\mathcal{A}_L$.*

Proof. Fix any set of dn items arriving in rounds $1, \ldots, dn$. The most profitable item v^* from this set arrives in the sampling phase with probability c/d. If this event occurs, no item in rounds $cn + 1, \ldots, dn$ beats v^* and $\mathcal{A}_L$ will not select any item. ◀

We finally note that our approach from Section 4.1 provides a general framework to obtain algorithms for 2-KS using secretary algorithms with two choices. Although stronger algorithms than Algorithm 2 exist for the 2-secretary objective [3, 12] and similar objectives [40, 42], it is not clear if they would improve the performance of the overall algorithm. More sophisticated algorithms may use weaker thresholds to accept the first item, which decreases the probability considered in Lemma 4.8. This, in turn, reduces the expected profit gained from $\mathcal{A}_S$, as described above.

5 Small Items

For small items, we use solutions for the fractional problem variant and obtain an integral packing via randomized rounding. This approach has been applied successfully to packing LPs [27]; however, for the knapsack problem it is not required to solve LP relaxations in each

round (as in [27]). Instead, here, we build upon solutions of the classical greedy algorithm, which is well-known to be optimal for the fractional knapsack problem. Particularly, this algorithm is both efficient in running time and easy to analyze.

We next formalize the greedy solution for any set T of items. Let the *density* of an item be the ratio of its profit to its size. Consider any list L containing the items from T ordered by non-increasing density. We define the *rank* $\rho(i)$ of item i as its position in L and $\sigma(l)$ as the item at position l in L. Thus, $\sigma(l) = \rho^{-1}(l)$ denotes the l-th densest item. Let k be such that $\sum_{i=1}^{k-1} s_{\sigma(i)} < W \leq \sum_{i=1}^{k} s_{\sigma(i)}$. The fraction of item i in the greedy solution α is now defined as

$$\alpha_i = \begin{cases} 1 & \text{if } \rho(i) < k \\ \left(W - \sum_{i=1}^{k-1} s_{\sigma(i)}\right) / s_i & \text{if } \rho(i) = k \\ 0 & \text{else}\,, \end{cases}$$

i.e., we pack the $k-1$ densest items integrally and fill the remaining space by the maximum feasible fraction of the k-th densest item. Let $\mathrm{OPT}(T)$ and $\mathrm{OPT}^*(T)$ denote the profits of optimal integral and fractional packings of T, respectively. It is not hard to see that α satisfies $\sum_{i \in T} \alpha_i v_i = \mathrm{OPT}^*(T) \geq \mathrm{OPT}(T)$ and $\sum_{i \in T} \alpha_i s_i = W$.

5.1 Algorithm

The algorithm $\mathcal{A}_S$ for small items, which is formally defined in Algorithm 3, works as follows. After a sampling phase of dn rounds, in each round $\ell \geq dn + 1$ the algorithm computes a greedy solution $x^{(\ell)}$ for $I_S(\ell)$. Here, $I_S(\ell)$ denotes the subset of I_S revealed up to round ℓ. The algorithm packs the current online item i with probability $x_i^{(\ell)}$. However, generally, this can only be done if the remaining capacity of the knapsack is at least $\delta W \geq s_i$.

Note that in case of an integral coefficient $x_i^{(\ell)} \in \{0, 1\}$, the packing step is completely deterministic. Moreover, in any greedy solution $x^{(\ell)}$, there is at most one item i with fractional coefficient $x_i^{(\ell)} \in (0, 1)$. Therefore, in expectation, there is only a small number of rounds where the algorithm actually requests randomness.

▶ **Observation 5.1.** *Let X denote the number of rounds where Algorithm 3 packs an item with probability $x_i \in (0, 1)$. It holds that $\mathbf{E}[X] \leq \ln(1/d) \leq 0.44$.*

Proof. Consider any round ℓ and let $x^{(\ell)}$ be the greedy knapsack solution computed by Algorithm 3. By definition of $x^{(\ell)}$, at most one of the ℓ visible items has a fractional coefficient $x_i^{(\ell)} \in (0, 1)$. The probability that this item i arrives in round ℓ is $1/\ell$ in a random permutation. Let X_ℓ be an indicator variable for the event that Algorithm 3 packs an item at random in round ℓ. By the above argument, we have $\mathbf{Pr}[X_\ell = 1] \leq 1/\ell$. Since Algorithm 3 selects items starting in round $dn+1$, we obtain $\mathbf{E}[X] = \sum_{\ell=dn+1}^{n} \mathbf{E}[X_\ell] \leq \sum_{\ell=dn+1}^{n} \frac{1}{\ell} \leq \ln \frac{1}{d} \leq 0.44\,.$ ◀

Note that Algorithm 2 and the sequential approach (Algorithm 1) are deterministic algorithms. Therefore, our overall algorithm requests randomness in expectation in less than one round.

5.2 Analysis

Let α be the greedy (offline) solution for I_S and set $\Delta = \frac{1}{1-\delta}$. Recall that in round $dn + 1$, the knapsack might already have been filled by $\mathcal{A}_L$ with large items in previous rounds. For now, we assume an empty knapsack after round dn and define this event as ξ. In the final analysis, we will use the fact that $\mathbf{Pr}[\xi]$ can be bounded from below, which is according to Lemma 4.8.

Algorithm 3 Algorithm $\mathcal{A}_S$ for small items.

Input : Random permutation of n $(1/3)$-small items, a knapsack of capacity W, parameter $d \in (0,1)$.
Output : A feasible (integral) packing of the knapsack.
Let ℓ be the current round and i be the online item of round ℓ.
if $\ell \leq dn$ **then**
  Sampling phase – discard all items.
if $dn+1 \leq \ell \leq n$ **then**
  Let $x^{(\ell)}$ be the greedy solution for $I_S(\ell)$.
  if *the remaining capacity is at least* δW **then**
    Pack i with probability $x_i^{(\ell)}$.

▶ **Lemma 5.2.** *Let $i \in I_S$ and $E_i(\ell)$ be the event that the item i is packed by $\mathcal{A}_S$ in round ℓ. For $\ell \geq dn+1$, it holds that* $\mathbf{Pr}\left[E_i(\ell) \mid \xi\right] \geq \frac{1}{n}\alpha_i(1-\Delta\ln\frac{\ell}{dn})$.

Proof. In a random permutation, item i arrives in round ℓ with probability $1/n$. In round $\ell \geq dn+1$, the algorithm decides to pack i with probability $x_i^{(\ell)}$. Note that the rank of item i in $I_S(\ell)$ is less or equal to its rank in I_S. According to the greedy solution's definition, this implies $x_i^{(\ell)} \geq \alpha_i$. Finally, the δ-small item i can be packed successfully if the current resource consumption X is at most $(1-\delta)W$. In the following, we investigate the expectation of X to give a probability bound using Markov's inequality at the end of this proof.

Let X_k be the resource consumption in round $k < \ell$. By assumption, the knapsack is empty after round dn, we have $X = \sum_{k=dn+1}^{\ell-1} X_k$. Let Q be the set of k visible items in round k. The set Q can be seen as uniformly drawn from all k-item subsets and any item $j \in Q$ is the current online item of round k with probability $1/k$. The algorithm packs any item j with probability $x_j^{(k)}$, thus

$$\mathbf{E}\left[X_k\right] = \sum_{j\in Q} \mathbf{Pr}\left[j \text{ occurs in round } k\right] s_j x_j^{(k)} = \frac{1}{k}\sum_{j\in Q} s_j x_j^{(k)} \leq \frac{W}{k}\,,$$

where the last inequality holds because $x^{(k)}$ is a feasible solution for a knapsack of size W. By the linearity of expectation and the previous equation, the expected resource consumption up to round ℓ is $\mathbf{E}\left[X\right] = \sum_{k=dn+1}^{\ell-1}\mathbf{E}\left[X_k\right] \leq \sum_{k=dn+1}^{\ell-1}\frac{W}{k} \leq W\ln\frac{\ell}{dn}$. Using Markov's inequality, we obtain finally

$$\mathbf{Pr}\left[X < (1-\delta)W\right] = 1 - \mathbf{Pr}\left[X \geq (1-\delta)W\right] \geq 1 - \frac{\mathbf{E}\left[X\right]}{(1-\delta)W} \geq 1 - \Delta\ln\frac{\ell}{dn}\,. \qquad \blacktriangleleft$$

Using Lemma 5.2 we easily obtain the total probability that a specific item will be packed.

▶ **Lemma 5.3.** *Let $i \in I_S$ and E_i be the event that the item i is packed by $\mathcal{A}_S$. It holds that* $\mathbf{Pr}\left[E_i \mid \xi\right] \geq \alpha_i\left((1-d)(1+\Delta) - \Delta\ln\frac{1}{d}\right)$.

Proof. Summing the probabilities from Lemma 5.2 over all rounds $\ell \geq dn+1$ gives

$$\begin{aligned}\mathbf{Pr}\left[E_i \mid \xi\right] &= \sum_{\ell=dn+1}^{n}\mathbf{Pr}\left[E_i(\ell)\mid\xi\right] \geq \sum_{\ell=dn+1}^{n}\frac{1}{n}\alpha_i\left(1-\Delta\ln\frac{\ell}{dn}\right)\\ &= \frac{1}{n}\alpha_i\left(n - dn - \Delta\sum_{\ell=dn+1}^{n}\ln\frac{\ell}{dn}\right) = \alpha_i\left(1-d-\frac{\Delta}{n}\sum_{\ell=dn+1}^{n}\ln\frac{\ell}{dn}\right).\end{aligned}$$

Since $\ln \frac{\ell}{dn}$ is monotonically increasing in ℓ, we can bound the last sum by the corresponding integral:

$$\sum_{\ell=dn+1}^{n} \ln \frac{\ell}{dn} \leq \int_{\ell=dn+1}^{n+1} \ln \frac{\ell}{dn} \, \mathrm{d}\ell = (n+1) \ln \frac{n+1}{dn} - (n+1) - (dn+1) \ln \frac{dn+1}{dn} + (dn+1) \, .$$

This implies $\lim_{n\to\infty} \frac{\Delta}{n} \sum_{\ell=dn+1}^{n} \ln \frac{\ell}{dn} \leq \Delta \left(\ln \frac{1}{d} - 1 + d \right)$. Rearranging terms gives the claim. ◀

The following lemma bounds the expected profit of the packing of $\mathcal{A}_S$, assuming ξ.

▶ **Lemma 5.4.** *It holds that* $\mathbf{E}\left[\mathcal{A}_S \mid \xi\right] \geq \left((1-d)(1+\Delta) - \Delta \ln \frac{1}{d}\right) \mathrm{OPT}_S$.

Proof. Let $\beta = (1-d)(1+\Delta) - \Delta \ln \frac{1}{d}$. By Lemma 5.3, the probability that an item i gets packed is $\mathbf{Pr}\left[E_i \mid \xi\right] \geq \alpha_i \beta$. Therefore,

$$\mathbf{E}\left[\mathcal{A}_S \mid \xi\right] = \sum_{i \in I_S} \mathbf{Pr}\left[E_i \mid \xi\right] v_i \geq \sum_{i \in I_S} \alpha_i \beta v_i \geq \beta \, \mathrm{OPT}_S \, . \qquad \blacktriangleleft$$

The conditioning on ξ can be resolved using Lemma 4.8. Thus we obtain the following lemma, which is the second pillar in the proof of Theorem 1.1 and concludes this section.

▶ **Lemma 5.5.** *With* $c = 0.42291$ *and* $d = 0.64570$*, we have* $\mathbf{E}\left[\mathcal{A}_S\right] \geq \frac{1}{6.65} \mathrm{OPT}_S$.

Proof. By Lemma 4.8, the probability for an empty knapsack after round dn is $\mathbf{Pr}\left[\xi\right] \geq \frac{c}{d}$. Thus, from Lemma 5.4 with $\Delta = \frac{1}{1-1/3} = \frac{3}{2}$, we obtain

$$\mathbf{E}\left[\mathcal{A}_S\right] = \mathbf{Pr}\left[\xi\right] \mathbf{E}\left[\mathcal{A}_S \mid \xi\right] = \frac{c}{d} \left(\frac{5}{2}(1-d) - \frac{3}{2} \ln \frac{1}{d} \right) \mathrm{OPT}_S \geq \frac{1}{6.65} \mathrm{OPT}_S \, . \qquad \blacktriangleleft$$

6 Extension to GAP

In this section we show that the sequential approach introduced in Section 3 can be easily adapted to GAP, yielding a (1/6.99)-competitive randomized algorithm. We first define the problem formally.

GAP. We are given a set of items $I = [n]$ and a set of resources $R = [m]$ of capacities $W_r \in \mathbb{Q}_{>0}$ for $r \in R$. If item $i \in I$ is assigned to resource $r \in R$, this raises profit (value) $v_{i,r} \in \mathbb{Q}_{\geq 0}$, but consumes $s_{i,r} \in \mathbb{Q}_{>0}$ of the resource's capacity. The goal is to assign each item to at most one resource such that the total profit is maximized and no resource exceeds its capacity. We call the tuple $(v_{i,r}, s_{i,r})$ an *option* of item i and w.l.o.g. assume that options for all resources exist. This can be ensured by introducing dummy options with $v_{i,r} = 0$. In the online version of the problem, in each round an item is revealed together with its set of options. The online algorithm must decide immediately and irrevocably, if the item is assigned. If so, it has to specify the resource according to one of its options.

Again, we construct restricted instances I_L and I_S according to the following definition, which generalizes Definition 2.1. Let $\delta \in (0, 1)$.

▶ **Definition 6.1.** *We call an option* $(v_{i,r}, s_{i,r})$ δ*-large if* $s_{i,r} > \delta W_r$ *and* δ*-small if* $s_{i,r} \leq \delta W_r$*. Whenever* δ *is clear from the context, we say an option is large or small for short. Based on a given instance* I *for GAP, we define two modified instances* I_L *and* I_S *which are obtained from* I *as follows.*

- I_L*: Replace each small option* $(v_{i,r}, s_{i,r})$ *by the large option* $(0, W_r)$*.*
- I_S*: Replace each large option* $(v_{i,r}, s_{i,r})$ *by the small option* $(0, \delta)$*.*

Thus, I_L only contains large options and I_S only contains small options. However, by construction no algorithm will assign an item according to a zero-profit option. We define OPT, OPT_L, and OPT_S accordingly. Note that the inequality $\mathrm{OPT} \leq \mathrm{OPT}_L + \mathrm{OPT}_S$ holds also for GAP.

The sequential framework of Algorithm 1 can be adapted in a straightforward manner by replacing terms like *packing* with *assignment to resource* r. Here, we set the threshold parameter to $\delta = 1/2$. In the following subsections, we specify algorithms $\mathcal{A}_L$ and $\mathcal{A}_S$ for $(1/2)$-large and $(1/2)$-small options, respectively.

6.1 Large Options

If each item consumes more than one half of a resource, no two items can be assigned to this resource. Thus, we obtain the following matching problem.

Edge-weighted bipartite matching problem. Given a bipartite graph $G = (L \cup R, E)$ and a weighting function $w\colon E \to \mathbb{Q}_{\geq 0}$, the goal is to find a bipartite matching $M \subseteq E$ such that $w(M) := \sum_{e \in M} w(e)$ is maximal. In the online version, the (offline) nodes from R and the number $n = |L|$ are known in advance, whereas the nodes from L are revealed online together with their incident edges. In the case of GAP, L is the set of items, R is the set of resources, and the weight of an edge $e = \{l, r\}$ is $w(e) = v_{l,r}$, i.e., the profit gained from assigning item l to resource r.

Under random arrival order, Kesselheim et al. [26] developed an optimal $(1/e)$-competitive algorithm for this problem. Adapting this algorithm to the sequential approach with parameters c and d leads to the following algorithm $\mathcal{A}_L$: After sampling the first cn nodes, in each round ℓ the algorithm computes a maximum edge-weighted matching $M^{(\ell)}$ for the graph revealed up to this round. Let $l \in L$ be the online vertex of round ℓ. If l is matched in $M^{(\ell)}$ to some node $r \in R$, we call $e^{(\ell)} = \{l, r\}$ the *tentative edge* of round ℓ. Now, if r is still unmatched and $\ell \leq dn$, the tentative edge is added to the matching.

A formal description of this algorithm is given in Appendix B.1. The proof of the approximation guarantee relies mainly on the following two lemmas; for completeness, we give the proofs from [26] in Appendix B.1. The first lemma shows that the expected weight of any tentative edge can be bounded from below.

▶ **Lemma 6.2** ([26]). *In any round ℓ, the tentative edge (if it exists) has expected weight* $\mathbf{E}\left[w(e^{(\ell)})\right] \geq \frac{1}{n}\,\mathrm{OPT}_L$.

However, we only gain the weight of the tentative edge $e^{(\ell)} = \{l, r\}$ if it can be added to the matching, i.e., if r has not been matched previously. The next lemma bounds the probability for this event from below.

▶ **Lemma 6.3** ([26]). *Let $\xi(r, \ell)$ be the event that the offline vertex $r \in R$ is unmatched after round ℓ. It holds that* $\mathbf{Pr}\left[\xi(r, \ell)\right] \geq \frac{cn}{\ell}$.

Using Lemmas 6.2 and 6.3, we can bound the competitive ratio of $\mathcal{A}_L$ in the following lemma. Note that we obtain the optimal algorithm from [26] for $c = 1/e$ and $d = 1$.

▶ **Lemma 6.4.** *For $n \to \infty$, it holds that* $\mathbf{E}\left[\mathcal{A}_L\right] \geq c \ln \frac{d}{c}\,\mathrm{OPT}_L$.

Proof. Let A_ℓ be the gain of the matching weight in round ℓ. As the tentative edge $e^{(\ell)} = \{l, r\}$ can only be added if r has not been matched in a previous round, we have $\mathbf{E}\left[A_\ell\right] = \mathbf{E}\left[w(e^{(\ell)})\right] \mathbf{Pr}\left[\xi(r, \ell)\right]$ for the event $\xi(r, \ell)$ from Lemma 6.3. Therefore, from

Lemmas 6.2 and 6.3 we have $\mathbf{E}[A_\ell] \geq \frac{1}{n}\operatorname{OPT}_L \frac{cn}{\ell} = \frac{c}{\ell}\operatorname{OPT}_L$. Summing over all rounds from $cn+1$ to dn yields

$$\mathbf{E}[\mathcal{A}_L] = \sum_{\ell=cn+1}^{dn} \mathbf{E}[A_\ell] \geq \left(c \sum_{\ell=cn+1}^{dn} \frac{1}{\ell}\right) \operatorname{OPT}_L \geq c \ln \frac{dn+1}{cn+1} \operatorname{OPT}_L \ .$$

Here, in the last step we used the fact $\sum_{\ell=cn+1}^{dn} \frac{1}{\ell} \geq \int_{cn+1}^{dn+1} \frac{1}{\ell}\,\mathrm{d}\ell = \ln \frac{dn+1}{cn+1}$. The claim follows by $\lim_{n\to\infty} \ln \frac{dn+1}{cn+1} = \ln \frac{d}{c}$. ◀

6.2 Small Options

For δ-small options we use the LP-based algorithm $\mathcal{A}_S$ from [27, Sec. 3.3]. On a high level, this algorithm works as follows: After a sampling phase of dn rounds, in each round ℓ the algorithm computes an optimal fractional solution for the instance revealed so far and uses the coefficients as probabilities for an integral assignment. In Appendix B.2 we prove the following lemma, where $\Delta = \frac{1}{1-\delta}$.

▶ **Lemma 6.5.** *For $n \to \infty$, it holds that $\mathbf{E}[\mathcal{A}_S] \geq \frac{c}{d}\left((1+\Delta)(1-d) - \Delta \ln \frac{1}{d}\right)\operatorname{OPT}_S$.*

Note that we obtain basically the same competitive ratio as in Lemma 5.4. Since Lemma 6.5 already addresses possible resource consumption due to assignments made by $\mathcal{A}_L$ in earlier rounds, the factor c/d arises (see Lemma 6.3).

6.3 Proof of Theorem 1.2

Finally, we prove our main theorem for GAP.

Proof of Theorem 1.2. We set the threshold between large and small options to $\delta = 1/2$ and consider Algorithm 1 with the algorithms $\mathcal{A}_L$ and $\mathcal{A}_S$ as defined previously. By Lemma 6.4, the expected gain of profit in rounds $cn+1, \ldots, dn$ is $\mathbf{E}[\mathcal{A}_L] \geq c \ln \frac{d}{c} \operatorname{OPT}_L$. Further, we gain $\mathbf{E}[\mathcal{A}_S] \geq \frac{c}{d}\left((1+\Delta)(1-d) - \Delta \ln \frac{1}{d}\right)\operatorname{OPT}_S$ with $\Delta = 2$ in the following rounds, according to Lemma 6.5. For parameters $c = 0.5261$ and $d = 0.6906$, we obtain $c \ln \frac{d}{c} \geq \frac{c}{d}\left(3(1-d) - 2\ln \frac{1}{d}\right)$ and thus, using $\operatorname{OPT}_L + \operatorname{OPT}_S \geq \operatorname{OPT}$,

$$\mathbf{E}[\mathcal{A}_L] + \mathbf{E}[\mathcal{A}_S] \geq \frac{c}{d}\left(3(1-d) - 2\ln \frac{1}{d}\right)(\operatorname{OPT}_L + \operatorname{OPT}_S) \geq \frac{1}{6.99}\operatorname{OPT} \ . \qquad ◀$$

References

1 Shipra Agrawal, Zizhuo Wang, and Yinyu Ye. A Dynamic Near-Optimal Algorithm for Online Linear Programming. *Operations Research*, 62(4):876–890, 2014.

2 Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. The Online Stochastic Generalized Assignment Problem. In *Proc. 16th International Workshop on Approximation, Randomization, and Combinatorial Optimization and 17th International Workshop on Randomization and Computation (APPROX/RANDOM)*, pages 11–25, 2013.

3 Susanne Albers and Leon Ladewig. New results for the k-secretary problem. Unpublished manuscript, 2018.

4 Moshe Babaioff, Jason Hartline, and Robert Kleinberg. Selling banner ads: Online algorithms with buyback. In *Fourth Workshop on Ad Auctions*, 2008.

5 Moshe Babaioff, Jason D. Hartline, and Robert D. Kleinberg. Selling ad campaigns: online algorithms with cancellations. In *Proc. 10th ACM Conference on Electronic Commerce (EC)*, pages 61–70, 2009.

6 Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. A Knapsack Secretary Problem with Applications. In *Proc. 10th International Workshop on Approximation, Randomization, and Combinatorial Optimization and 11th International Workshop on Randomization and Computation (APPROX/RANDOM)*, pages 16–28, 2007.

7 Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. Matroid Secretary Problems. *Journal of the ACM*, 65(6):35:1–35:26, 2018.

8 Bahman Bahmani, Aranyak Mehta, and Rajeev Motwani. A 1.43-Competitive Online Graph Edge Coloring Algorithm in the Random Order Arrival Model. In *Proc. 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 31–39, 2010.

9 Christian Borgs, Jennifer T. Chayes, Nicole Immorlica, Kamal Jain, Omid Etesami, and Mohammad Mahdian. Dynamics of bid optimization in online advertisement auctions. In *Proc. 16th International Conference on World Wide Web (WWW)*, pages 531–540, 2007.

10 Niv Buchbinder and Joseph Naor. Online Primal-Dual Algorithms for Covering and Packing. *Math. Oper. Res.*, 34(2):270–286, 2009.

11 Dirk G. Cattrysse and Luk N. Van Wassenhove. A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research*, 60(3):260–272, 1992.

12 T.-H. Hubert Chan, Fei Chen, and Shaofeng H.-C. 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 *Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1169–1188, 2015.

13 Chandra Chekuri and Sanjeev Khanna. A Polynomial Time Approximation Scheme for the Multiple Knapsack Problem. *SIAM Journal on Computing (SICOMP)*, 35(3):713–728, 2005.

14 Henrik I. Christensen, Arindam Khan, Sebastian Pokutta, and Prasad Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79, 2017.

15 Eugene B Dynkin. The optimum choice of the instant for stopping a Markov process. *Soviet Mathematics*, 4:627–629, 1963.

16 Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S. Mirrokni, and Clifford Stein. Online Stochastic Packing Applied to Display Ad Allocation. In *Proc. 18th Annual European Symposium on Algorithms (ESA)*, pages 182–194, 2010.

17 Jon Feldman, Nitish Korula, Vahab S. Mirrokni, S. Muthukrishnan, and Martin Pál. Online Ad Assignment with Free Disposal. In *Proc. 5th International Workshop Internet and Network Economics (WINE)*, pages 374–385, 2009.

18 P.R. Freeman. The secretary problem and its extensions: A review. *International Statistical Review/Revue Internationale de Statistique*, pages 189–206, 1983.

19 Waldo Gálvez, Fabrizio Grandoni, Sandy Heydrich, Salvatore Ingala, Arindam Khan, and Andreas Wiese. Approximating Geometric Knapsack via L-Packings. In *Proc. 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 260–271, 2017.

20 Oliver Göbel, Thomas Kesselheim, and Andreas Tönnis. Online Appointment Scheduling in the Random Order Model. In *Proc. 23rd Annual European Symposium on Algorithms (ESA)*, pages 680–692, 2015.

21 Xin Han, Yasushi Kawase, and Kazuhisa Makino. Online Unweighted Knapsack Problem with Removal Cost. *Algorithmica*, 70(1):76–91, 2014.

22 Xin Han, Yasushi Kawase, and Kazuhisa Makino. Randomized algorithms for online knapsack problems. *Theoretical Computer Science*, 562:395–405, 2015.

23 Kazuo Iwama and Shiro Taketomi. Removable Online Knapsack Problems. In *Proc. 29th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 293–305, 2002.

24 Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems*. Springer, 2004.

25 Claire Kenyon. Best-Fit Bin-Packing with Random Order. In *Proc. 7th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 359–364, 1996.

26 Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. An Optimal Online Algorithm for Weighted Bipartite Matching and Extensions to Combinatorial Auctions. In *Proc. 21st Annual European Symposium on Algorithms (ESA)*, pages 589–600, 2013.

27 Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. Primal Beats Dual on Online Packing LPs in the Random-Order Model. *SIAM J. Comput.*, 47(5):1939–1964, 2018.

28 Samir Khuller, Stephen G. Mitchell, and Vijay V. Vazirani. On-Line Algorithms for Weighted Bipartite Matching and Stable Marriages. *Theoretical Computer Science*, 127(2):255–267, 1994.

29 Robert D. Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *Proc. 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 630–631, 2005.

30 Nitish Korula, Vahab S. Mirrokni, and Morteza Zadimoghaddam. Online Submodular Welfare Maximization: Greedy Beats 1/2 in Random Order. *SIAM J. Comput.*, 47(3):1056–1086, 2018.

31 Denis V Lindley. Dynamic programming and decision theory. *Applied Statistics*, pages 39–51, 1961.

32 George S. Lueker. Average-Case Analysis of Off-Line and On-Line Knapsack Problems. *J. Algorithms*, 29(2):277–305, 1998.

33 Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing LPs. In *Proc. 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 597–606, 2011.

34 Alberto Marchetti-Spaccamela and Carlo Vercellis. Stochastic on-line knapsack problems. *Mathematical Programming*, 68:73–104, 1995.

35 Silvano Martello and Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Inc., New York, NY, USA, 1990.

36 Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. AdWords and generalized online matching. *Journal of the ACM*, 54(5):22, 2007.

37 Adam Meyerson. Online Facility Location. In *Proc. 42nd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 426–431, 2001.

38 Vahab S. Mirrokni, Shayan Oveis Gharan, and Morteza Zadimoghaddam. Simultaneous approximations for adversarial and stochastic online budgeted allocation. In *Proc. 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1690–1701, 2012.

39 Marco Molinaro and R. Ravi. The Geometry of Online Packing Linear Programs. *Math. Oper. Res.*, 39(1):46–59, 2014.

40 ML Nikolaev. On a generalization of the best choice problem. *Theory of Probability & Its Applications*, 22(1):187–190, 1977.

41 Temel Öncan. A Survey of the Generalized Assignment Problem and Its Applications. *Information Systems and Operational Research INFOR*, 45(3):123–141, 2007.

42 Mitsushi Tamaki. Recognizing both the maximum and the second maximum of a sequence. *Journal of Applied Probability*, 16(4):803–812, 1979.

43 Rahul Vaze. Online knapsack problem and budgeted truthful bipartite matching. In *Proc. IEEE Conference on Computer Communications (INFOCOM) 2017*, pages 1–9, 2017.

44 Rahul Vaze. Online Knapsack Problem Under Expected Capacity Constraint. In *Proc. IEEE Conference on Computer Communications (INFOCOM) 2018*, pages 2159–2167, 2018.

45 Yunhong Zhou, Deeparnab Chakrabarty, and Rajan M. Lukose. Budget Constrained Bidding in Keyword Auctions and Online Knapsack Problems. In *Proc. 4th International Workshop Internet and Network Economics (WINE)*, pages 566–576, 2008.

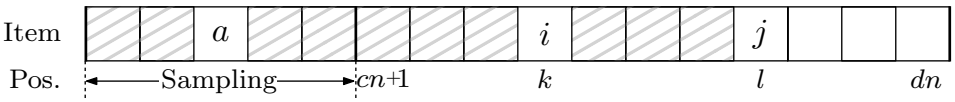

Figure 2 Input sequence considered in Lemma 4.3. The gray dashed slots represent items of rank greater than a.

A Missing Proofs for the Knapsack Result

Proof of Lemma 4.3. Let $i \in [n-1]$ and $j = i+1$. The proof follows the same structure as the proof of Lemma 4.2. Again, we construct the permutation by drawing the positions for items i, j, a first and afterwards all remaining items with position up to $\mathrm{pos}(j)$. Fix positions $k = \mathrm{pos}(i)$ and $l = \mathrm{pos}(j)$. Again, $\mathrm{pos}(a) \le cn$ must hold by definition of a. The probability that a random permutation satisfies these three position constraints is $\beta := \frac{1}{n}\frac{1}{n-1}\frac{cn}{n-2}$. All remaining items up to position l must have rank greater than a (see Figure 2). Thus we need to draw $l-3$ items from a set of $n-3$ remaining items, from which $n-a$ have rank greater than a. This happens with probability $h(n-3, n-a, l-3)$. Using the law of total probability for $cn+1 \le k < l \le dn$ and $a \in \{j+1, \dots, n\}$, we obtain

$$\begin{aligned} p_{ij} &= \beta \sum_{k=cn+1}^{dn-1} \sum_{l=k+1}^{dn} \sum_{a=j+1}^{n} h(n-3, n-a, l-3) \\ &= \beta \sum_{k=cn+1}^{dn-1} \sum_{l=k+1}^{dn} \frac{1}{\binom{n-3}{l-3}} \sum_{a=j+1}^{n} \binom{n-a}{l-3} = \beta \sum_{k=cn+1}^{dn-1} \sum_{l=k+1}^{dn} \frac{\binom{n-j}{l-2}}{\binom{n-3}{l-3}}, \end{aligned}$$

where in the last step we used the equality $\sum_{a=j+1}^{n} \binom{n-a}{l-3} = \sum_{a=0}^{n-j-1} \binom{a}{l-3} = \binom{n-j}{l-2}$.

We next consider the asymptotic setting $n \to \infty$. For this purpose, we define $Q(l) = \binom{n-j}{l-2}/\binom{n-3}{l-3}$. For $(i,j) = (1,2)$ we have $Q(l) = \binom{n-2}{l-2}/\binom{n-3}{l-3} = \frac{n-2}{l-2}$. The sum $\sum_{l=k+1}^{dn} \frac{n-2}{l-2}$ converges to $n \ln \frac{dn}{k}$ for $n \to \infty$. Further, $\lim_{n\to\infty} \sum_{k=cn+1}^{dn-1} n \ln \frac{dn}{k} = n\,(F(dn) - F(cn))$ for $F(x) := x \ln \frac{dn}{x} + x$. Hence,

$$\lim_{n\to\infty} p_{12} = \lim_{n\to\infty} \beta n \left(dn \ln \frac{dn}{dn} + dn - cn \ln \frac{dn}{cn} - cn \right) = c \left(d - c \ln \frac{d}{c} - c \right).$$

In the case $(i,j) = (2,3)$ it holds that $Q(l) = \binom{n-3}{l-2}/\binom{n-3}{l-3} = \frac{n-l}{l-2}$ and we have $\lim_{n\to\infty} \sum_{l=k+1}^{dn} \frac{n-l}{l-2} = n \ln \frac{dn}{k} - dn + k$. Let $F(x) := nx\left(\ln \frac{dn}{x} - d + 1\right) + \frac{x^2}{2}$. Again, by bounding the sum by the corresponding integral we obtain

$$\begin{aligned} &\lim_{n\to\infty} \sum_{k=cn+1}^{dn} n \ln \frac{dn}{k} - dn + k \\ &= F(dn) - F(cn) \\ &= dn^2 \left(\ln \frac{dn}{dn} - d + 1 \right) + \frac{d^2 n^2}{2} - cn^2 \left(\ln \frac{dn}{cn} - d + 1 \right) - \frac{c^2 n^2}{2} \\ &= n^2 \left(-d^2 + d + \frac{d^2}{2} - c \ln \frac{d}{c} + cd - c - \frac{c^2}{2} \right) \\ &= n^2 \left(d - c \ln \frac{d}{c} - c - \frac{d^2}{2} + cd - \frac{c^2}{2} \right). \end{aligned}$$

Multiplying the last term with $\lim_{n\to\infty} \beta = c/n^2$ gives the claim for p_{23}. ◀

Proof of Lemma 4.4. Suppose i is accepted first and j is accepted as the second item in the input sequence π. Consider the sequence π' obtained from π by swapping i with j. Since j and i are the first two elements beating the best sampling item in π', Algorithm 2 will select j and i on input π'. Hence, the number of permutations must be the same for both events, which implies the claim. ◀

Proof of Lemma 4.5. The argument is similar to the proof of Lemma 4.4. Consider any input sequence π where i is selected first and j second. We know that the best item a from sampling has profit $v_a < v_j < v_i$ and thus any item k with $i < k < j$ must occur after j. Let π' be the sequence obtained from π by swapping i with k. Now, i is behind k and j, thus Algorithm 2 will accept k and j. Again, this proves $p_{ij} = p_{kj}$ since the numbers of corresponding permutations are equal. ◀

The next lemma is used in the proof of Lemma 4.7 to show that for the given lists of parameters, we have $\beta_w \geq \gamma_w$.

▶ **Lemma A.1.** *Let $f(x) = 2\ln x - 6x + 2x^2 - \frac{x^3}{3}$. For parameters c,d with $f(c) \geq f(d)$ it holds that $\beta_w \geq \gamma_w$ where $2 \leq w \leq 5$.*

Proof. The function f is chosen in a way that $f(c) \geq f(d)$ is equivalent to $\beta_5 \geq \gamma_5$. This can be verified easily, using $\beta_5 = p_{\mathrm{D}} = p_1$, $\gamma_5 = p_{\mathrm{E}} + p_{\mathrm{F}} + p_{\mathrm{G}} = p_2 + p_3 + p_4$, and Lemma 4.2. Therefore, the claim for $w = 5$ holds by assumption. For $2 \leq w \leq 4$, the claims follow immediately from $f(c) \geq f(d)$ and the symmetry property of Lemma 4.4:

$$\beta_2 = p_{\mathrm{H}} = p_1 - p_{12} = p_1 - p_{21} \geq p_2 - p_{21} = p_{\mathrm{J}} = \gamma_2$$
$$\beta_3 = p_{\mathrm{I}} = p_1 - p_{13} = p_1 - p_{31} \geq p_2 + p_3 - p_{31} = p_{\mathrm{E}} + p_{\mathrm{L}} = \gamma_3$$
$$\beta_4 = p_{\mathrm{D}} + p_K = p_1 + p_2 - p_{23} \geq p_1 - p_{32} \geq p_3 - p_{32} = p_{\mathrm{M}} = \gamma_4\,. \qquad ◀$$

B Missing Proofs for the GAP Result

B.1 Large Options

Algorithm 4 Algorithm for edge-weighted bipartite matching from [26] (extended by our parameters c, d).

Input : Offline vertex set R, number of online vertices $n = |L|$,
parameters $c, d \in (0, 1)$ with $c < d$.
Output : Matching M.
Set $M = \emptyset$.
Let ℓ be the current round and l be the online vertex of round ℓ.
if $1 \leq \ell \leq cn$ **then**
  Sampling phase – do not add any edge.
if $cn + 1 \leq \ell \leq dn$ **then**
  Let $M^{(\ell)}$ be a maximum-weight matching for the graph in round ℓ.
  Let $e^{(\ell)} \in M^{(\ell)}$ be the edge incident to l.
  if $M \cup e^{(\ell)}$ *is a matching* **then**
    Add $e^{(\ell)}$ to M.
  if $\ell > dn$ **then**
    Do not add any edge.

Proof of Lemma 6.2. Let $e^{(\ell)}$ be the tentative edge of round ℓ and let $Q \subseteq L$ with $|Q| = \ell$ be the set of visible vertices from this round. Since each vertex from Q has the same probability of $1/\ell$ to arrive in round ℓ, we have

$$\mathbf{E}\left[w(e^{(\ell)})\right] = \sum_{e=\{l,r\}\in M^{(\ell)}} \mathbf{Pr}\left[l \text{ arrives in round } \ell\right] w(e) = \frac{1}{\ell} w(M^{(\ell)})\,. \tag{4}$$

Let $M^* = M^{(n)}$ be a maximum weight (offline) matching and $M^*_Q = \{e = \{l,r\} \in M^* \mid l \in Q\}$. We have $w(M^{(\ell)}) \geq w(M^*_Q)$, since $M^{(\ell)}$ is an optimal and M^*_Q a feasible matching for the graph revealed in round ℓ. As Q can be seen as uniformly drawn among all ℓ-element subsets, each vertex l has probability ℓ/n to be in Q. It follows

$$\mathbf{E}\left[w(M^{(\ell)})\right] \geq \mathbf{E}\left[w(M^*_Q)\right] = \sum_{e=\{l,r\}\in M^*} \mathbf{Pr}\left[l \in Q\right] w(e) = \frac{\ell}{n} w(M^*)\,. \tag{5}$$

Combining (4) and (5) concludes the proof. ◀

Proof of Lemma 6.3. In each round k, the vertex r can only be matched if it is incident to the tentative edge $e^{(k)} \in M^{(k)}$ of this round, i.e., $e^{(k)} = \{l,r\}$ where $l \in L$ is the online vertex of round k. As l can be seen as uniformly drawn among all k visible nodes (particularly, independent from the order of the previous $k-1$ items), l has probability $1/k$ to arrive in round k. Consequently, r is not matched in round k with probability $1 - 1/k$. This argument applies to all rounds $cn+1, \ldots, \ell$. Therefore,

$$\mathbf{Pr}\left[\xi(r,\ell)\right] \geq \prod_{k=cn+1}^{\ell} 1 - \frac{1}{k} = \prod_{k=cn+1}^{\ell} \frac{k-1}{k} = \frac{cn}{\ell}\,. \qquad ◀$$

B.2 Small Options

For δ-small options we use the LP-based algorithm from [27, Sec. 3.3] and analyze it within our algorithmic framework. In order to make this paper self-contained, we give a linear program for GAP (LP 1), the algorithm, and its corresponding proofs.

$$\begin{aligned}
\text{maximize} \quad & \sum_{\substack{i\in I_S \\ r\in R}} v_{i,r} x_{i,r} & & \\
\text{subject to} \quad & \sum_{i\in I_S} s_{i,r} x_{i,r} \leq W_r & & \forall r \in R \\
& \sum_{r\in R} x_{i,r} \leq 1 & & \forall i \in I_S \\
& x_{i,r} \in \{0,1\} & & \forall (i,r) \in I_S \times R
\end{aligned} \tag{LP 1}$$

Let $\mathcal{A}_S$ be Algorithm 5. After a sampling phase of dn rounds, in each round ℓ the algorithm computes an optimal solution $x^{(\ell)}$ of the relaxation of LP 1 for $I_S(\ell)$. Here, $I_S(\ell)$ denotes the instance of small options revealed so far. Now, the decision to which resource the current online item i is assigned, if at all, is made by randomized rounding using $x^{(\ell)}$: Resource $r \in R$ is chosen with probability $x^{(\ell)}_{i,r}$ and the item stays unassigned with probability $1 - \sum_{r\in R} x^{(\ell)}_{i,r}$. Note that it is only feasible to assign the item to the chosen resource if its remaining capacity is at least δW_r.

Algorithm 5 GAP algorithm for small options from [27, Sec. 3.3].

Input : Random order sequence of small options,
parameter $d \in (0,1)$.
Output : Integral GAP assignment.
Let ℓ be the current round and i be the online item of round ℓ.
if $1 \leq \ell \leq dn$ **then**
  Sampling phase – do not assign any item.
if $dn+1 \leq \ell \leq n$ **then**
  Let $x^{(\ell)}$ be an optimal fractional solution of LP 1 for $I_S(\ell)$.
  Choose a resource r (possibly none), where r has probability $x^{(\ell)}_{i,r}$.
  if *the remaining capacity of* r *is at least* δW_r **then**
    Assign i to r.

To analyze Algorithm 5, we consider the gain of profit in round $\ell \geq dn+1$, denoted by A_ℓ. For this purpose, let $i^{(\ell)}$ be the item of that round and $r^{(\ell)}$ the resource chosen by the algorithm. Now, it holds that $\mathbf{E}[A_\ell] = \mathbf{E}\left[v_{i^{(\ell)},r^{(\ell)}}\right] \mathbf{Pr}\left[i^{(\ell)} \text{ can be assigned to } r^{(\ell)}\right]$, where in the first term, the expectation is over the item arriving in round ℓ and the resource chosen by the algorithm. The latter term only depends on the resource consumption of $r^{(\ell)}$ in earlier rounds. In the next two lemmas we give lower bounds for both terms.

▶ **Lemma B.1** ([27, Sec. 3.3]). *For any round $\ell \geq dn+1$, it holds that $\mathbf{E}\left[v_{i^{(\ell)},r^{(\ell)}}\right] \geq \frac{1}{n}\,\mathrm{OPT}_S$.*

Proof. The proof is similar to Lemma 6.2. As we consider a fixed round ℓ, we write i and r instead of $i^{(\ell)}$ and $r^{(\ell)}$ for ease of presentation. Further, we write $v(\alpha) := \sum_{j \in I_S} \sum_{s \in R} \alpha_{j,s} v_{j,s}$ for the profit of a fractional assignment α.

Fix any set Q of ℓ visible items in round ℓ. Let $x^{(n)}$ be an optimal (offline) solution to the relaxation of LP 1. Further, let $x^{(n)}|_Q$ denote the restriction of $x^{(n)}$ to the items in Q, i.e., $(x^{(n)}|_Q)_{j,s} = x^{(n)}_{j,s}$ if $j \in Q$ and $(x^{(n)}|_Q)_{j,s} = 0$ if $j \notin Q$. Since $x^{(n)}|_Q$ is a feasible and $x^{(\ell)}$ is an optimal solution for Q, we have $\mathbf{E}\left[v(x^{(\ell)})\right] \geq \mathbf{E}\left[v(x^{(n)}|_Q)\right]$. As in a random permutation each item has the same probability of ℓ/n to be in Q, it holds that

$$\mathbf{E}\left[v(x^{(\ell)})\right] \geq \mathbf{E}\left[v(x^{(n)}|_Q)\right] = \sum_{j \in I_S} \sum_{s \in R} \mathbf{Pr}[j \in Q]\, x^{(n)}_{j,s} v_{j,s} = \frac{\ell}{n} v(x^{(n)}) = \frac{\ell}{n}\,\mathrm{OPT}_S\,. \quad (6)$$

Similarly, each item from Q is the current online item i with probability $1/\ell$. The resource s, to which an item j gets assigned, is determined by randomized rounding using $x^{(\ell)}_{j,s}$. Therefore we get

$$\mathbf{E}[v_{i,r}] = \sum_{j \in Q} \sum_{s \in R} \mathbf{Pr}[j = i, s = r]\, v_{j,s} = \sum_{j \in Q} \sum_{s \in R} \frac{1}{\ell} x^{(\ell)}_{j,s} v_{j,s} = \frac{1}{\ell} v(x^{(\ell)})\,. \quad (7)$$

Combining (6) and (7) gives the claim. ◀

Hence, by the previous lemma the expected gain of profit in each round is a $(1/n)$-fraction of OPT_S, supposing the remaining resource capacity is large enough. The probability for the latter event is considered in the following lemma. Here, a crucial property is that we deal with δ-small options. Let $\Delta = \frac{1}{1-\delta}$.

▶ **Lemma B.2.** *For any round $\ell \geq dn+1$, we have $\mathbf{Pr}\left[i^{(\ell)} \text{ can be assigned to } r^{(\ell)}\right] \geq \frac{c}{d}\left(1 - \Delta \ln \frac{\ell}{dn}\right)$.*

Proof. Let ξ be the event that no item is assigned to r after round dn. Note that ξ does not necessarily hold, since $\mathcal{A}_L$ might already have assigned items to r in earlier rounds. By Lemma 6.3, $\mathbf{Pr}[\xi] \geq \frac{c}{d}$. Therefore, it remains to show $\mathbf{Pr}\left[i^{(\ell)} \text{ can be assigned to } r^{(\ell)} \mid \xi\right] \geq 1 - \Delta \ln \frac{\ell}{dn}$.

For this purpose, assume that ξ holds and let X denote the resource consumption of r after round $\ell - 1$. Further, let X_k be the resource consumption of r in round $k < \ell$. We have $X = \sum_{k=dn+1}^{\ell-1} X_k$. Let Q be the set of k visible items in round k. The set Q can be seen as uniformly drawn from all k-item subsets and any item $j \in Q$ is the current online item of round k with probability $1/k$. Now, the algorithm assigns any item j to resource r with probability $x_{j,r}^{(k)}$, thus

$$\mathbf{E}[X_k] = \sum_{j \in Q} \mathbf{Pr}[j \text{ occurs in round } k]\, s_{j,r} x_{j,r}^{(k)} = \frac{1}{k} \sum_{j \in Q} s_{j,r} x_{j,r}^{(k)} \leq \frac{W_r}{k}\,, \tag{8}$$

where the last inequality follows from the capacity constraint for resource r in LP 1. By linearity of expectation and inequality (8), the expected resource consumption up to round ℓ is thus

$$\mathbf{E}[X] = \sum_{k=dn+1}^{\ell-1} \mathbf{E}[X_k] \leq \sum_{k=dn+1}^{\ell-1} \frac{W_r}{k} \leq W_r \ln \frac{\ell}{dn}\,. \tag{9}$$

Now, since $i^{(\ell)}$ is δ-small, $X < (1-\delta)W_r$ implies $X + s_{i^{(\ell)},r^{(\ell)}} \leq W_r$ in which case the assignment is feasible. Using (9) and Markov's inequality, we obtain

$$\mathbf{Pr}[X < (1-\delta)W_r] = 1 - \mathbf{Pr}[X \geq (1-\delta)W_r] \geq 1 - \frac{\mathbf{E}[X]}{(1-\delta)W_r} \geq 1 - \Delta \ln \frac{\ell}{dn}\,. \qquad \blacktriangleleft$$

Now, the bound on the competitive ratio of $\mathcal{A}_S$ from Lemma 6.5 follows.

Proof of Lemma 6.5. We add the expected profits in single rounds using Lemmas B.1 and B.2.

$$\begin{aligned}
\mathbf{E}[\mathcal{A}_S] &= \sum_{\ell=dn+1}^{n} \mathbf{E}[A_\ell] = \sum_{\ell=dn+1}^{n} \mathbf{E}\left[v_{i^{(\ell)},r^{(\ell)}}\right] \mathbf{Pr}\left[i^{(\ell)} \text{ can be assigned to } r^{(\ell)}\right] \\
&\geq \sum_{\ell=dn+1}^{n} \frac{1}{n} \operatorname{OPT}_S \frac{c}{d} \left(1 - \Delta \ln \frac{\ell}{dn}\right) = \frac{c}{dn} \left(\sum_{\ell=dn+1}^{n} 1 - \Delta \ln \frac{\ell}{dn}\right) \operatorname{OPT}_S \\
&= \frac{c}{dn} \left(n - dn - \Delta \sum_{\ell=dn+1}^{n} \ln \frac{\ell}{dn}\right) \operatorname{OPT}_S\,.
\end{aligned}$$

Since $\frac{\ell}{dn}$ is monotone increasing in ℓ, we have $\sum_{\ell=dn+1}^{n} \ln \frac{\ell}{dn} \leq \int_{dn+1}^{n+1} \ln \frac{\ell}{dn}\, \mathrm{d}\ell$ and this integral evaluates to $(n+1)\ln\frac{n+1}{dn+1} - (n+1) - (dn+1)\ln\frac{dn+1}{dn} + (dn+1)$. For $n \to \infty$, this approaches $n \ln \frac{1}{d} - n + dn$. Hence, we have $\lim_{n\to\infty} \mathbf{E}[\mathcal{A}_S] \geq \frac{c}{d}\left((1+\Delta)(1-d) - \Delta \ln \frac{1}{d}\right) \operatorname{OPT}_S$. $\blacktriangleleft$