PACKING A KNAPSACK OF UNKNOWN CAPACITY*

YANN DISSER[†], MAX KLIMM[‡], NICOLE MEGOW[§], AND SEBASTIAN STILLER[¶]

Abstract. We study the problem of packing a knapsack without knowing its capacity. Whenever we attempt to pack an item that does not fit, the item is discarded; if the item fits, we have to include it in the packing. We show that there is always a policy that packs a value within factor 2 of the optimum packing, irrespective of the actual capacity. If all items have unit density, we achieve a factor equal to the golden ratio $\varphi \approx 1.618$. Both factors are shown to be best possible.

In fact, we obtain the above factors using packing policies that are *universal* in the sense that they fix a particular order of the items in the beginning and try to pack the items in this order, without changing the order later on. We give efficient algorithms computing these policies. On the other hand, we show that, for any $\alpha > 1$, the problem of deciding whether a given universal policy achieves a factor of α is coNP-complete. If α is part of the input, the same problem is shown to be coNP-complete for items with unit densities. Finally, we show that it is coNP-hard to decide, for given α, whether a set of items admits a universal policy with factor α, even if all items have unit densities.

Key words. Knapsack, unknown capacity, robustness, approximation guarantees

AMS subject classifications. 68W25, 68W27, 68Q25

1. Introduction.

In the standard knapsack problem we are given a set of items, each associated with a size and a value, and a capacity of the knapsack. The goal is to find a subset of the items with maximum value whose size does not exceed the capacity. In this paper, we study a variant of the knapsack problem where the capacity of the knapsack is not given. Whenever we try to pack an item, we observe whether or not it fits the knapsack. If it does, the item is packed into the knapsack and cannot be removed later. If it does not fit, we discard it and continue packing with the remaining items. We call the problem *knapsack problem with unknown capacity.* The central question of this paper is how much we lose by not knowing the capacity, in the worst case.

A solution to the knapsack problem with unknown capacity is a policy that governs the order in which we attempt to pack the items, depending only on the observation which of the previously attempted items did fit into the knapsack and which did not. In other words, a policy is a binary decision tree with the item that is tried first at its root. The two children of the root are the items that are tried next, which of the two depends on whether or not the first item fits the knapsack, and so on. We aim for a solution that is good for *every* possible capacity, compared to the best solution of the standard knapsack problem for this capacity. Formally, a policy has *robustness factor* α if, for any capacity, packing according to the policy results in a value that is

*An extended abstract with parts of this work appeared in the proceedings of STACS 2014 [13]. The first author was supported by the Alexander von Humboldt Foundation. The research of the second author was carried out in the framework of MATHEON supported by Einstein Foundation Berlin. The third author was supported by the German Science Foundation (DFG) under contract ME 3825/1.

[†]Department of Mathematics, Technische Universität Darmstadt, Dolivostraße 15, 64293 Darmstadt, Germany. (disser@mathematik.tu-darmstadt.de).

[‡]Department of Mathematics, Technische Universität Berlin, Straße des 17. Juni 136, 10623 Berlin, Germany. (klimm@math.tu-berlin.de).

[§]Department of Mathematics, Technische Universität München, Boltzmannstraße 3, 85747 Garching bei München, Germany. (nmegow@ma.tum.de).

[¶]Department of Mathematics, Technische Universität Braunschweig, Pockelstraße 14, 38106 Braunschweig, Germany. (sebastian.stiller@tu-braunschweig.de).

at least a $1/\alpha$-fraction of the optimum value for this capacity.

Direct applications of the knapsack problem with unknown capacity include settings where the capacity remains uncertain until it is (nearly) exhausted. For example, this may be the case when mining natural ressources and serving orders for different quantities before the ressource is depleted, or when cutting steel plates of given sizes from steel coils of varying lengths. The capacity-oblivious variant of the knapsack problem also naturally arises whenever items are prioritized by a different entity or at a different time than the actual packing of the knapsack. This is for instance the case in settings where cargo is loaded onto a vessel with varying remaining capacity, in case we cannot expect the loading personnel to reoptimize on the fly and, instead, have to provide a policy before knowing the capacity. Recently, parts of our results where applied to a different model related to the optimization of energy consumption in mobile telecommunication [12].

1.1. Our results. We show that the knapsack problem with unknown capacity always admits a robustness factor of 2. In fact, this robustness factor can be achieved with a policy that packs the items according to a fixed order, irrespective of the observations made while packing. Such a policy is called *universal.* We provide an algorithm that computes a 2-robust, universal policy in time $\Theta(n \log n)$ for a given set of n items. We complement this result by showing that no robustness factor better than 2 can be achieved in general, even by policies that are not universal. In other words, the cost of not knowing the capacity is exactly 2.

We give a different efficient algorithm for the case that all items have unit density, i.e., size and value of each item coincide. This algorithm produces a universal policy with a robustness factor of at most the golden ratio $\varphi \approx 1.618$. Again, we show that no better robustness factor can be achieved in general, even by policies that are not universal.

While good universal policies can be found efficiently, it is intractable to compute the robustness factor of a *given* universal policy and it is intractable to compute the best robustness factor an instance admits. Specifically, we show that, for any fixed $\alpha \in (1, \infty)$, it is coNP-complete to decide whether a given universal policy is α-robust. For unit densities we establish a slightly weaker hardness result by showing that it is coNP-complete to decide whether a given universal policy achieves a *given* robustness factor α. Finally, we show that, for given α, it is coNP-hard to decide whether an instance of the knapsack problem with unknown capacity admits a universal policy with robustness factor α, even when all items have unit density.

1.2. Related work. The knapsack problem has been studied for various models of imperfect information. In the majority of the studied models, the lack of full information concerns the items and their arrival but not the knapsack capacity.

Marchetti-Spaccamela and Vercellis [31] introduced the *online* knapsack problem in which the knapsack capacity is known and items arrive online one by one. When an item is presented, it must be accepted or rejected before the next item arrives. In this seminal paper it is shown that the problem in its full generality does not admit online algorithms with a guaranteed profit within a constant of the offline optimum solution. Various problem variants have been studied since then and non-trivial bounds have been derived. Examples include online knapsack with resource augmentation (Iwama and Zhang [24]), the removable online knapsack problem (Iwama and Taketomi [23], Han et al. [20, 19, 18]), the online partially fractional knapsack problem ([36]), items arriving in a random order (Babaioff et al. [1]), the stochastic online knapsack problem (Marchetti-Spaccamela and Vercellis [31], Kleywegt and Papastavrou [27, 28], van

Slyke and Young [40]) and online knapsack with advice (Böckenhauer et al. [5]).

In the *stochastic* knapsack problem, the set of items is known but sizes and values of the items are random variables. It is known that a policy maximizing the expected value is PSPACE-hard to compute, see Dean et al. [10]. The authors assume that the packing stops when the first item does not fit the knapsack, and give a universal policy that approximates the value obtained by an optimal, not necessarily universal, policy by a factor of 2. Bhalgat et al. [4] complement this result by giving a universal PTAS for the case that the knapsack capacity may be violated by a factor of $1 + \epsilon$.

In *robust* knapsack problems, a set of possible scenarios for the sizes and values of the items is given. Yu [43], Bertsimas and Sim [3], Goetzmann et al. [17], and Monaci and Pferschy [35] study the problem of maximizing the worst-case value of a knapsack under various models. Büsing et al. [7] and Bouman et al. [6] study the problem from a computational point of view. Both allow for an adjustment of the solution after the realization of the scenario. Similar to our model, Bouman et al. [6] consider uncertainty in the capacity.

The notion of a *robustness factor* that we adopt in this work is due to Hassin and Rubinstein [22] and is defined as the worst-case ratio of solution and optimum, over all realizations. Kakimura et al. [26] analyze the complexity of deciding whether an α-robust solution exists for a knapsack instance with an unknown bound on the number of items that can be packed. Recently, Kobayashi and Takazawa [29] studied randomized strategies for this setting.

Megow and Mestre [33] study a variant of the knapsack problem with unknown capacity closely related to ours. In contrast to our model, they assume that the packing stops once the first item does not fit the remaining capacity. In this model, no algorithm can guarantee to achieve a constant robustness factor, and, thus, the authors resort to *instance-sensitive* performance guarantees. They provide a PTAS that constructs a universal policy with robustness factor arbitrarily close to the best possible robustness factor for every particular instance. Diodati et al. [12] propose to add to this model the mild assumption that no item size exceeds the unknown knapsack capacity. Interestingly, they achieve results very similar to ours in the model that allows to discard non-fitting items. While our lower bounds (given in our extended abstract [13]) apply to their model, Diodati et al. [12] also give a best-possible 2-robust algorithm.

The *incremental* knapsack problem is another related problem that has been studied by Hartline and Sharp [21]. The key difference to our model is that the different possible knapsack capacities are known in advance and that their number is constant. The authors give an FPTAS for approximating the optimal robustness factor for the special case of proportional values. Thielen et al. [41] investigate an online variant of the incremental knapsack problem in which in each time period new items arrive online and the knapsack capacity increases incrementally. They present deterministic and randomized upper and lower bounds on the competitive ratio as a function of the time horizon.

The concept of *universal solutions* is used in various other contexts (explicitly or implicitly), such as hashing (Carter and Wegman [8]), caching (Frigo et al. [15], Bender et al. [2]), routing (Valiant and Brebner [42], Räcke [38]), TSP (Papadimitriou [37], Deineko et al. [11], Jia et al. [25]), Steiner tree and set cover (Jia et al. [25]), matching (Hassin and Rubinstein [22], Matuschke et al. [32]), and scheduling (Epstein et al. [14], Megow and Mestre [33]). In all of these works, the general idea is that specific parameters of a problem instance are unknown, e.g., the cache size or the set of vertices to visit in a TSP tour, and the goal is to find a universal solution that performs well for all realizations of the hidden parameters.

Universal policies for the knapsack problem with unknown capacity play a role in the design of public key cryptosystems. One of the first such systems – the Merkle-Hellman knapsack cryptosystem [34] – is based on particular instances that allow for a 1-robust universal policy for this knapsack variant. The basic version of this cryptosystem can be attacked efficiently, e.g., by the famous attack of Shamir [39]. This attack uses the fact that the underlying knapsack instance has exponentially increasing item sizes. A better understanding of universal policies may help to develop knapsack-based cryptosystems that avoid the weaknesses of Merkle and Hellman's.

2. Preliminaries. An instance of the *knapsack problem with unknown capacity* is given by a set of n items $\mathcal{I}$, where each item $i \in \mathcal{I}$ has a non-negative *value* $v(i) \in \mathbb{Q}_{\geq 0}$ and a strictly positive *size* $l(i) \in \mathbb{Q}_{>0}$. For a subset $S \subseteq \mathcal{I}$ of items, we write $v(S) = \sum_{i \in S} v(i)$ and $l(S) = \sum_{i \in S} l(i)$ to denote its total value and total size, respectively, of the items in S. A *solution* for instance $\mathcal{I}$ is a policy $\mathcal{P}$ that governs the order in which the items are considered for packing into the knapsack. The policy must be independent of the capacity of the knapsack, but the choice which item to try next may depend on the observations which items did and which items did not fit the knapsack so far. Formally, a solution policy is a binary decision tree that contains every item exactly once along each path from the root to a leaf. The *packing* $\mathcal{P}(C) \subseteq \mathcal{I}$ of $\mathcal{P}$ for a fixed capacity C is obtained as follows: We start with an empty knapsack $X = \emptyset$ and check whether the item r at the root of $\mathcal{P}$ fits the knapsack, i.e., whether $l(r) + l(X) \leq C$. If the item fits, we add r to X and continue packing recursively with the left subtree of r. Otherwise, we discard r and continue packing recursively with the right subtree of r. Once we reached a leaf, we set $\mathcal{P}(C) = X$.

A *universal policy* Π for instance $\mathcal{I}$ is a policy that does not depend on observations made while packing, i.e., the decision tree for a universal policy has a fixed permutation of the items along every path from the root to a leaf. We identify a universal policy with this fixed permutation and write $\Pi = (\Pi_1, \Pi_2, \ldots, \Pi_n)$. Analogously to general policies, the packing $\Pi(C) \subseteq \mathcal{I}$ of a universal policy Π for capacity $C \leq l(\mathcal{I})$ is obtained by considering the items in the order given by the permutation Π and adding every item if it does not exceed the remaining capacity. We measure the quality of a policy for the knapsack problem with unknown capacity by comparing its packing with the optimal packing for each capacity. More precisely, a policy $\mathcal{P}$ for instance $\mathcal{I}$ is called *α-robust for capacity C*, $\alpha \geq 1$, if it holds that $v(\textsc{Opt}(\mathcal{I}, C)) \leq \alpha \cdot v(\mathcal{P}(C))$, where $\textsc{Opt}(\mathcal{I}, C)$ denotes an optimal packing for capacity C. We say $\mathcal{P}$ is *α-robust* if it is α-robust for all capacities. In this case, we call α the *robustness factor* of policy $\mathcal{P}$.

3. Solving the Knapsack Problem with Unknown Capacity. In this section, we describe an efficient algorithm that constructs a universal policy for a given instance of the knapsack problem with unknown capacity. The solution produced by our algorithm is guaranteed to pack at least half the value of the optimal solution for any capacity C. We show that this is the best possible robustness factor.

The analysis of our algorithm relies on the classical *modified greedy* algorithm (cf. [30]). We compare the packing of our policy, for each capacity, to the packing obtained by the modified greedy algorithm instead of the actual optimum. As the modified greedy is a 2-approximation, to show that our policy is 2-robust it is sufficient to show that its packing is never worse than the one obtained by the modified greedy algorithm. We briefly review the modified greedy algorithm.

Let $d(i) = v(i)/l(i)$ denote the *density* of item i. The modified greedy algorithm

Algorithm 1 MGREEDY($\mathcal{I}$, C)

Input: set of items $\mathcal{I}$, capacity C
Output: subset $S \subseteq \mathcal{I}$ such that $l(S) \leq C$ and $v(S) \geq v(\text{OPT}(\mathcal{I}, C))/2$
$D \leftarrow \langle$items in $\{i \in \mathcal{I} \mid l(i) \leq C\}$ sorted non-increasingle by density $d\rangle$
$k \leftarrow \max\{j \mid l(\{D_1, \dots, D_j\}) \leq C\}$
$P \leftarrow (D_1, \dots, D_k)$, $s \leftarrow D_{k+1}$
if $v(P) \geq v(s)$ **then**
  return P
else
  return $\{s\}$
end if

(MGREEDY) for a set of items $\mathcal{I}$ and known knapsack capacity C first discards all items that are larger than C from $\mathcal{I}$. The remaining items are sorted in non-increasing order of their densities, breaking ties arbitrarily. The algorithm then either takes the longest prefix P of the resulting sequence that still fits into capacity C, or the first item s that does not fit anymore, depending on which of the two has a greater value, see Algorithm 1 for a formal description.

We evaluate the quality of our universal policy by comparing it for every capacity with the solution of MGREEDY. This analysis suffices because of the following well-known property of the modified greedy algorithm.

THEOREM 1 (cf. [30]). *For every instance $(\mathcal{I}, C)$ of the standard knapsack problem with known capacity, $v(\text{OPT}(\mathcal{I}, C)) \leq 2 \cdot v(\text{MGREEDY}(\mathcal{I}, C))$.*

For our analysis, it is helpful to fix the tie-breaking rule under which MGREEDY initially sorts the items. To this end, we assume that there is a bijection $t : \mathcal{I} \to \{1, 2, \dots, n\}$, that maps every item $i \in \mathcal{I}$ to a *tie-breaking index* $t(i)$, and that the modified greedy algorithm initially sorts the items decreasingly with respect to the tuple $\tilde{d}(\cdot) = (d(\cdot), t(\cdot))$, i.e., the items are sorted non-increasingly by density and whenever two items have the same density, they are sorted by decreasing tie-breaking index. In the following, for two items i, j, we write $\tilde{d}(i) \succ \tilde{d}(j)$ if and only if $d(i) > d(j)$, or $d(i) = d(j)$ and $t(i) > t(j)$, and say that i has higher density than j.

We are now ready to describe our algorithm UNIVERSAL (Algorithm 2) that produces a universal policy inspired by the the behavior of MGREEDY but with the crucial difference that the capacity is unknown. Our algorithm starts with an empty permutation and then inserts items at specific places in the permutation. When inserting items into the permutation, we use the following wording. Let $\Pi = (\Pi_1, \dots, \Pi_k)$ be a permutation of k items and let i be an item not contained in Π. When we say that we insert item i *directly in front of item Π_j this means that after the insertion, the permutation is $\Pi' = (\Pi_1, \dots, \Pi_{j-1}, i, \Pi_j, \dots, \Pi_k)$. In contrast, after inserting item i* in front of all items*, the permutation is $\Pi' = (i, \Pi_1, \dots, \Pi_k)$. For a permutation $\Pi = (\Pi_1, \dots, \Pi_k)$, we also say that item Π_j, $j \in \{1, \dots, k-1\}$ is* directly in front of *item Π_{j+1}, and that item Π_{j+1} is* directly behind *item Π_j. We also say that the items $\Pi_1, \dots, \Pi_{j-1}$ are* in front of *item Π_j and the items $\Pi_{j+1}, \dots, \Pi_k$ are behind item Π_j.*

To get some intuition for our universal algorithm, recall that for a given capacity, MGREEDY has to make the choice between taking the maximum prefix in the density-order or a single item of greater value. For a different capacity, the prefix will only be shorter/longer but the single item might be a completely different one. Now, the key to our universal algorithm is that we identify all items which might be a crucial

Algorithm 2 Universal($\mathcal{I}$)

Input: set of items $\mathcal{I}$
Output: sequence of items Π

$L \leftarrow \langle$items in $\mathcal{I}$ sorted by non-decreasing size$\rangle$
$\Pi^{(0)} \leftarrow \emptyset$
for $r \leftarrow 1, \ldots, n$ **do**
  if L_r is a swap item **then**
    $\Pi^{(r)} \leftarrow (L_r, \Pi^{(r-1)})$
  else
    $j \leftarrow 1$
    while $j \leq |\Pi^{(r-1)}|$ **and** $\tilde{d}(\Pi_j^{(r-1)}) \succ \tilde{d}(L_r)$ **do**
      $j \leftarrow j + 1$
    end while
    $\Pi^{(r)} \leftarrow (\Pi_1^{(r-1)}, \ldots, \Pi_{j-1}^{(r-1)}, L_r, \Pi_j^{(r-1)}, \ldots)$
  end if
end for
return $\Pi^{(n)}$

single item for some capacity. We call an item i *swap item* if it is worth more than all denser items that are not larger than i. Formally, we define it as follows.

Definition 2 (Swap Item). *Item i is a swap item if and only if*

$$v(i) > v(\{j \in \mathcal{I} \mid l(j) \leq l(i) \text{ and } \tilde{d}(j) > \tilde{d}_i\}).$$

Note that whenever MGreedy for a given capacity and a given tie-breaking rule chooses a single item instead of the prefix of densest items, then this item is a swap item as defined above.

Our algorithm, Universal, works as follows. First, we identify all swap items. Then we start with an empty permutation and consider all items for insertion in order of non-decreasing sizes. We place a swap item in front of all items that are already in the permutation, and we place any other item directly in front of the first item in the permutation that has a lower density; see Algorithm 2.

While it is important for our analysis that ties between items of equal density are broken according to the fixed tie-breaking rule given by $\tilde{d}$, it does not matter how ties are handled between items of equal size.

We prove the following result.

Theorem 3. *The algorithm* Universal *constructs a universal policy of robustness factor* 2.

Before we prove this theorem, we analyze the structure of the permutation produced by Universal in terms of density, size, and value. First, we prove that the item directly behind a non-swap item Π_k has lower density than Π_k.

Lemma 4. *For a sequence Π returned by* Universal*, we have $\tilde{d}(\Pi_k) \succ \tilde{d}(\Pi_{k+1})$ for every non-swap item Π_k, $1 \leq k < n$.*

Proof. For $j \in \{k, k+1\}$, let $r(j) \in \{1, \ldots, n\}$ be the index of the iteration in which Universal inserts Π_j into Π. We distinguish two cases.

If $r(k) < r(k+1)$, then the item Π_{k+1} cannot be a swap item, since it would appear in front of the item Π_k if it was. As each non-swap item is inserted into Π such that all items in front of it are larger with respect to $\tilde{d}$, the claim follows.

If $r(k) > r(k+1)$, since it is not a swap item, Π_k is put in front of Π_{k+1} because it has a higher density. □

We prove that no item in front of a swap item Π_k has smaller size than Π_k.

LEMMA 5. *For a permutation* Π *returned by* UNIVERSAL, *we have* $l(\Pi_j) \geq l(\Pi_k)$ *for every swap item* $\Pi_k, 1 < k \leq n$, *and every other item* $\Pi_j, 1 \leq j < k$.

Proof. Since Π_k is a swap item, it stands in front of all items inserted earlier into Π. Hence, all items that appear in front of Π_k in Π have been inserted in a later iteration of UNIVERSAL. Since UNIVERSAL processes items in order of non-decreasing sizes, we have $l(\Pi_j) \geq l(\Pi_k)$. □

We prove that no item in front of a swap item Π_k has smaller value than Π_k.

LEMMA 6. *For a permutation* Π *returned by* UNIVERSAL, *we have* $v(\Pi_j) \geq v(\Pi_k)$ *for every swap item* $\Pi_k, 1 < k \leq n$, *and every other item* $\Pi_j, 1 \leq j < k$.

Proof. We distinguish three cases.

First case: Π_j is a swap item and $\tilde{d}(\Pi_j) \succ \tilde{d}(\Pi_k)$. By Lemma 5, we have $l(\Pi_j) \geq l(\Pi_k)$, and the claim trivially holds.

Second case: Π_j is a swap item and $\tilde{d}(\Pi_j) \prec \tilde{d}(\Pi_k)$. Since Π_j is a swap item,

$$v(\Pi_j) > v(\{i \in \mathcal{I} \mid l(i) \leq l(\Pi_j) \text{ and } \tilde{d}(i) \succ \tilde{d}(\Pi_j)\}). \tag{1}$$

Since, by Lemma 5, $l(\Pi_j) \geq l(\Pi_k)$, the item Π_k is included in the set on the right hand side of (1). We conclude that $v(\Pi_j) \geq v(\Pi_k)$.

Third case: Π_j is not a swap item. Let $\Pi_{j'}$ be the first swap item behind Π_j in Π, i.e.,

$$j' = \min\{i \in \{j+1, \dots, k\} \mid \Pi_i \text{ is a swap item }\}.$$

Note that the minimum is well-defined as Π_k is a swap item. The analysis of the first two cases implies that $v(\Pi_{j'}) \geq v(\Pi_k)$. By Lemma 4 we have $\tilde{d}(\Pi_j) \succ \tilde{d}(\Pi_{j+1}) \succ \cdots \succ \tilde{d}(\Pi_{j'})$, and by Lemma 5 we have $l(\Pi_j) \geq l(\Pi_{j'})$. Hence, $v(\Pi_j) \geq v(\Pi_{j'}) \geq v(\Pi_k)$. □

Finally, the next lemma gives a legitimation for the violation of the density order in the output permutation. Essentially, whenever an item is in front of denser items, we guarantee that it is worth at least as much as all of them combined.

LEMMA 7. *For a permutation* Π *returned by* UNIVERSAL, *we have*

$$v(\Pi_k) \geq v\big(\{\Pi_j \mid j > k \text{ and } \tilde{d}(\Pi_j) \succ \tilde{d}(\Pi_k)\}\big)$$

for every item $\Pi_k, 1 \leq k < n$.

Proof. We distinguish whether Π_k is a swap item, or not.

First case: Π_k *is a swap item.* By definition,

$$v(\Pi_k) > v\big(\{\Pi_j \mid l(\Pi_j) \leq l(\Pi_k) \text{ and } \tilde{d}(\Pi_j) \succ \tilde{d}(\Pi_k)\}\big).$$

Since items whose size is strictly larger than $l(\Pi_k)$ are inserted into Π at a later iteration of UNIVERSAL, they can only end up behind Π_k if they are smaller with respect to $\tilde{d}$. Hence,

$$\{\Pi_j \mid j > k \text{ and } \tilde{d}(\Pi_j) \succ \tilde{d}(\Pi_k)\} \subseteq \{\Pi_j \mid l(\Pi_j) \leq l(\Pi_k) \text{ and } \tilde{d}(\Pi_j) \succ \tilde{d}(\Pi_k)\},$$

and thus $v(\Pi_k) > v(\{\Pi_j \mid j > k \text{ and } \tilde{d}(\Pi_j) \succ \tilde{d}(\Pi_k)\})$, as claimed.

Second case: Π_k is not a swap item. let $\Pi_{k'}$ be the first swap item behind it in Π. If no such item exists, the claim holds by Lemma 4, since

$$\big\{\Pi_j \mid j > k \text{ and } \tilde{d}(\Pi_j) \succ \tilde{d}(\Pi_k)\big\} = \emptyset.$$

Otherwise, by Lemma 4, we obtain $\tilde{d}(\Pi_k) \succ \tilde{d}(\Pi_{k+1}) \succ \cdots \succ \tilde{d}(\Pi_{k'})$ and hence

$$\begin{aligned}\{\Pi_j \mid j > k \text{ and } \tilde{d}(\Pi_j) \succ \tilde{d}(\Pi_k)\} &= \{\Pi_j \mid j > k' \text{ and } \tilde{d}(\Pi_j) \succ \tilde{d}(\Pi_k)\}\\ &\subseteq \{\Pi_j \mid j > k' \text{ and } \tilde{d}(\Pi_j) \succ \tilde{d}(\Pi_{k'})\}.\end{aligned}$$

Consequently, and by the argument above for swap items,

$$\begin{aligned}v(\Pi_{k'}) &> v(\{\Pi_j \mid j > k' \text{ and } \tilde{d}(\Pi_j) \succ \tilde{d}(\Pi_{k'})\})\\ &\geq v(\{\Pi_j \mid j > k \text{ and } \tilde{d}(\Pi_j) > \tilde{d}(\Pi_k)\})).\end{aligned}$$

Finally, by Lemma 6, we have $v(\Pi_k) \geq v(\Pi_{k'}) \geq v(\{\Pi_j \mid j > k \text{ and } \tilde{d}(\Pi_j) \succ \tilde{d}(\Pi_k)\})$. ∎

We now prove Theorem 3.

Proof (Theorem 3). We show that for every item set $\mathcal{I}$, the permutation Π returned by UNIVERSAL satisfies $v(\text{OPT}(\mathcal{I}, C)) \leq 2v(\Pi(C))$ for every capacity $C \leq l(\mathcal{I})$. By Theorem 1, it suffices to show $v(\Pi(C)) \geq v(\text{MGREEDY}(\mathcal{I}, C))$ for all capacities. We distinguish between capacities for which MGREEDY outputs the maximal prefix of the densest items that fits the capacity, and capacities for which MGREEDY outputs the first item after this prefix. We proceed to distinguish these two cases.

First case: MGREEDY outputs the maximal prefix of the densest items that still fits the capacity. Let $G^+ = \text{MGREEDY}(\mathcal{I}, C) \setminus \Pi(C)$ be the set of items packed by MGREEDY for capacity C that are not packed by the permutation Π. Similarly, let $U^+ = \Pi(C) \setminus \text{MGREEDY}(\mathcal{I}, C)$. If $G^+ = \emptyset$, then $v(\Pi(C)) \geq v(\text{MGREEDY}(\mathcal{I}, C))$ and we are done. Suppose now that $G^+ \neq \emptyset$. Then, also $U^+ \neq \emptyset$, since $\Pi(C)$ is inclusion maximal. For all items $i \in U^+$, we have $l(i) \leq C$ and $i \notin \text{MGREEDY}(\mathcal{I}, C)$. As $\text{MGREEDY}(\mathcal{I}, C)$ is a maximal prefix of the densest items for capacity C, we have $\tilde{d}(i) \prec \tilde{d}(i')$ for all $i \in U^+$ and $i' \in G^+$. By definition of $\Pi(C)$ and since $U^+ \neq \emptyset$, we also have $k = \min\{j \mid \Pi_j \in U^+\} < \min\{k' \mid \Pi_{k'} \in G^+\}$, i.e., the first item $\Pi_k \in U^+$ in Π is encountered before every item from G^+. It follows that

$$G^+ \subseteq \big\{\Pi_j \mid j > k \text{ and } \tilde{d}(\Pi_j) \succ \tilde{d}(\Pi_k)\big)\big\}.$$

By Lemma 7, $v(\Pi_k) \geq v(G^+)$, and hence we obtain

$$\begin{aligned}v(\Pi(C)) &= v\big(\Pi(C) \cap \text{MGREEDY}(\mathcal{I}, C)\big) + v(U^+)\\ &\geq v\big(\Pi(C) \cap \text{MGREEDY}(\mathcal{I}, C)\big) + v(\Pi_k)\\ &\geq v\big(\Pi(C) \cap \text{MGREEDY}(\mathcal{I}, C)\big) + v(G^+) = v(\text{MGREEDY}(\mathcal{I}, C)).\end{aligned}$$

Second case: MGREEDY outputs the first item after the maximal prefix of the densest items. Let $\{\Pi_k\} = \text{MGREEDY}(\mathcal{I}, C)$ be item returned by the modified greedy algorithm. Then, $\Pi(C)$ contains at least one item Π_j with $j \leq k$. If $j = k$, then trivially $v(\Pi(C)) \geq v(\text{MGREEDY}(\mathcal{I}, C))$. Otherwise, by Lemma 6, we have $v(\Pi(C)) \geq v(\Pi_j) \geq v(\Pi_k) = v(\text{MGREEDY}(\mathcal{I}, C))$. ∎

While it is obvious that UNIVERSAL runs in polynomial time, we show that it can be modified to run in time $\Theta(n \log n)$.

THEOREM 8. *The algorithm* UNIVERSAL *(Algorithm 2) can be implemented to run in time* $\Theta(n \log n)$.

Proof. In a first phase the algorithm identifies all swap items, in a second phase it constructs the output permutation Π. We show that each phase can be implemented to run in time $\Theta(n \log n)$.

For the first phase, recall that an item is a swap item, if and only if it is worth more than all smaller items of higher density combined. To determine all swap-items, we first sort the items decreasing by density. Then we insert the items in this order into a balanced search tree which itself is ordered by size. As additional information, in each tree node j we store the total value of items in both subtrees below j. While traversing the search tree to insert an item j this additional information allows to calculate the sum of values of all smaller and denser (i.e., already inserted) items. Thus, by inserting an item into the tree we can determine whether it is a swap item. Sorting, inserting and updating the additional information takes $\Theta(n \log n)$ time.

We construct the output permutation Π by iterating over the items in order of increasing size, as in Algorithm 2. We maintain a list Λ of balanced search trees, each ordered by density. Except for the last tree in Λ, every tree contains exactly one swap item, which is the item of the smallest density in the tree. The density of a tree is the density of this swap item (or 0 if the tree has no swap item). Each tree stores the items in Π in front of the corresponding swap item (if it exists) and behind the swap item of the preceding tree in Λ (if it exists). We start with a list containing a single tree with no corresponding swap item, which eventually holds all non-swap items that end up behind the last swap item in Π. Whenever we encounter a new swap item, we add a new tree consisting of only this swap item to the front of Λ. For each non-swap item, we have to find the correct tree to insert it into. Once we know the tree, we can determine the position at which to insert the item into the tree, and thus in Π, in time $\Theta(\log n)$ simply by searching the tree.

To complete the proof, we need an efficient way to find the correct tree in Λ for a non-swap item. For this purpose, we maintain a sublist Λ' of Λ that contains only those trees that are needed for the remainder of the algorithm. Whenever a new swap item s adds a tree to the front of Λ, we also add the tree to the front of Λ'. Observe that from this point on no items are inserted into trees of a higher density than s. Hence, before inserting the tree of s to Λ', we may remove trees of higher density from the front of Λ'. This guarantees that Λ' remains sorted by density. We can thus implement Λ' as a balanced search tree ordered by density. This way, we can find the correct tree for each non-swap item in time $\Theta(\log n)$. Since every tree is removed at most once from Λ', the amortized cost for maintaining the sublist is constant for each swap item.

Since UNIVERSAL requires n iterations, the total running time is $\Theta(n \log n)$. ∎

The running time of our algorithm is best-possible in the sense that we can use it for sorting a set of n elements at a running time that is best possible for comparison-based algorithms; see, e.g. [9].

THEOREM 9. *Every algorithm that computes a universal policy with constant robustness factor* $\alpha \geq 2$ *can be used as a sorting algorithm with the same running time.*

Proof. Fix $\alpha \geq 2$ arbitrarily. For a given set of n unique non-negative integers $a_1, a_2, \dots, a_n$ to be sorted, we construct (in linear time) an instance of the knapsack problem with unknown capacity. For each integer a_i, $i \in \{1, \dots, n\}$, we have an item of size and value $l(i) = v(i) = \alpha^{2a_i}$. Notice that the exponential increase in the encoding length does not affect the running time of a universal policy as we make the standard assumption of the arithmetic running time model that arithmetic operations can be performed in constant time. In this model, the running time depends only on n.

It suffices to show that an α-robust universal policy for this instance must place elements in decreasing order of sizes. To that end, we consider capacities $l(i)$, $i \in \{1,\dots,n\}$, and show that for each of these capacities, the corresponding item i must be in the knapsack since no other subset has sufficiently large total value. For any $i \in \{1,\dots,n\}$, let $S(i) = \{i' \mid l(i') < l(i)\}$ denote the set of all items smaller than item i, and let i^* be the item with maximum size in $S(i)$. Then, $2a_{i*}+1 \le 2a_i - 1$ and thus

$$\sum_{i' \in S_i} v(i') = \sum_{i' \in S_i} \alpha^{2a_{i'}} \le \sum_{j=0}^{2a_{i*}} \alpha^j = \frac{\alpha^{2a_{i*}+1}-1}{\alpha - 1} < \alpha^{2a_i - 1} = \frac{v(i)}{\alpha}.$$

Any α-robust algorithm thus needs to ensure that item i is indeed in the knapsack for capacity $l(i)$, and, hence, item i must be in front every item in $S(i)$ in its universal solution. Since this must hold for every item i, the universal solution must be sorted decreasingly. In other words, we can directly deduce the solution for the sorting problem from an α-robust universal solution. ∎

We now give a general lower bound on the robustness factor of any policy for the knapsack problem with unknown capacity. This shows that UNIVERSAL is best possible in terms of robustness factor and running time in the sense of Theorem 9.

THEOREM 10. *For every $\delta > 0$, there are instances of the knapsack problem with unknown capacity where no policy achieves a robustness factor of $2 - \delta$.*

Proof. We give a family of instances, one for each size $n \ge 3$. We ensure that for every item i of the instance of size n, there is a capacity C, such that packing item i first can only lead to a solution that is worse than $\textsc{Opt}(\mathcal{I}, C)$ by a factor of at least $(2 - 4/n)$. This completes the proof, as the factor approaches 2 for increasing values of n. The instance of size n is given by $\mathcal{I} = \{1, 2, \dots, n\}$ with $l(i) = F_n + F_i - 1$, and $v(i) = 1 + \frac{i}{n}$, where F_i denotes the i-th Fibonacci number ($F_1 = 1, F_2 = 1, F_3 = 2, \dots$).

We need to show that, no matter which item is tried first (i.e., no matter which item is the root of the policy), there is a capacity for which this choice ruins the solution. Observe that both values and sizes of the items are strictly increasing. Assume that item $i \ge 3$ is packed first. Since the smallest item has size $l(1) = F_n$, for capacity $C_i = 2F_n + F_i - 2 < 2F_n + F_i - 1 = l(1) + l(i)$, no additional item fits the knapsack. However, the unique optimum solution in this case is $\textsc{Opt}(\mathcal{I}, C_i) = \{i-1, i-2\}$. These two items fit the knapsack, as $l(i-1) + l(i-2) = 2F_n + F_{i-1} + F_{i-2} - 2 = 2F_n + F_i - 2 = C_i$. By definition,

$$\frac{v(i-1) + v(i-2)}{v(i)} = \frac{2n + 2i - 3}{n + i} = 2 - \frac{3}{n+i} \ge 2 - \frac{3}{n}\,.$$

Thus, policies that first pack item $i \ge 3$ cannot attain a robustness factor better than $2 - 3/n$.

Now, assume that one of the two smallest items is packed first. For capacity $C_{1,2} = l(n) = 2F_n - 1 < 2F_n = l(1) + l(2)$, no additional item fits the knapsack. The unique optimum solution, however, is to pack item n. It remains to compute the ratios

$$\frac{v(n)}{v(1)} > \frac{v(n)}{v(2)} = \frac{2n}{n+2} = 2 - \frac{4}{n+2} > 2 - \frac{4}{n}\,.$$

Algorithm 3 UniversalUD($\mathcal{I}$)

Input: set of items $\mathcal{I}$
Output: sequence of items Π

$L \leftarrow \langle$items in $\mathcal{I}$ sorted such that $L_1 \prec \cdots \prec L_n\rangle$
$\Pi^{(0)} \leftarrow \emptyset$
for $r \leftarrow 1, \dots, n$ **do**
  $j \leftarrow 1$
  while $j \leq |\Pi^{(r-1)}|$ **and** $v(L_r) < \varphi v(\Pi_j^{(r-1)})$ **do**
    $j \leftarrow j + 1$
  end while
  $\Pi^{(r)} \leftarrow (\Pi_1^{(r-1)}, \dots, \Pi_{j-1}^{(r-1)}, L_r, \Pi_j^{(r-1)}, \dots)$
end for
return $\Pi^{(n)}$

Hence, policies that first pack item 1 or item 2 do not achieve a robustness factor better than $2 - 4/n$. ∎

4. Unit Densities. In this section we restrict ourselves to instances of the oblivious knapsack problem, where all items have unit density, i.e., $v(i) = l(i)$ for all items $i \in \mathcal{I}$. For two items $i, j \in \mathcal{I}$ we say that i is smaller than j and write $i \prec j$ if $v(i) < v(j)$, or $v(i) = v(j)$ and $t(i) < t(j)$, where t is the tie-breaking index introduced in Section 3. We give an algorithm UniversalUD (cf. Algorithm 3) that produces a universal policy tailored to achieve the best possible robustness factor equal to the golden ratio $\varphi \approx 1.618$. The algorithm considers the items from the smallest to the largest, and inserts each item into the output sequence as far to the end as possible, such that the item is not preceded by other items that are more than a factor φ smaller. Intuitively, the algorithm tries as much as possible to keep the resulting order sorted increasingly by size; only when an item dominates another item by a factor of at least φ the algorithm ensures that it precedes this item in the final sequence. Note that, even though φ is irrational, for rationals a, b the condition $a < \varphi b$ can be tested efficiently by testing the equivalent condition $a/b < 1 + b/a$.

Theorem 11. *The algorithm* UniversalUD *constructs a universal policy of robustness factor φ when all items have unit density.*

Proof. Given an instance $\mathcal{I}$ of the knapsack problem with unknown capacity with unit densities and any capacity $C \leq v(\mathcal{I})$, we compare the packing $\Pi(C)$ that results from the solution $\Pi = \text{UniversalUD}(\mathcal{I})$ with an optimal packing $\text{Opt}(\mathcal{I}, C)$. We define the set M of items in $\Pi(C)$ for which at least one smaller item is not in $\Pi(C)$, i.e., more precisely, let $M = \{i \in \Pi(C) \mid \exists j \in \mathcal{I} \backslash \Pi(C) : j \prec i\}$.

We first consider the case that $M \neq \emptyset$ and set $i = \min_{\prec} M$ to be the smallest item in M with respect to '$\prec$'. Consider the iteration r of UniversalUD in which i is inserted into Π, i.e., $i = L_r$. By definition of M, there is an item $j \prec i$ with $j \notin \Pi(C)$. Let j be the first such item in Π. Since $j \prec i$, we have $j \in \Pi^{(r)}$. From $i \in \Pi(C)$ and $j \notin \Pi(C)$, it follows that i precedes j in Π (and thus in $\Pi^{(r)}$). Let i' be the item directly preceding j in $\Pi^{(r)}$. If $i' = i$, i was compared with j when it was inserted into $\Pi^{(r)}$, with the result that $v(i) \geq \varphi v(j)$ and thus $v(\Pi(C)) \geq \varphi v(j)$. If $i' \neq i$, by definition of j, we still have $i' \in \Pi(C)$. Also, either $i' \succ j$ and thus $v(i') \geq v(j)$, or j was compared with i' when it was inserted into Π in an earlier iteration of UniversalUD, with the result that $v(i') > \frac{1}{\varphi} v(j)$. Again, $v(\Pi(C)) \geq v(i) + v(i') > v(j) + \frac{1}{\varphi} v(j) = \varphi v(j)$.

In both cases it follows from $j \notin \Pi(C)$ that $v(\text{OPT}(\mathcal{I}, C)) \leq C < v(\Pi(C)) + v(j)$, and using $v(j) \leq \frac{1}{\varphi} v(\Pi(C))$ we get

$$\frac{v(\text{OPT}(\mathcal{I}, C))}{v(\Pi(C))} < \frac{v(\Pi(C)) + v(j)}{v(\Pi(C))} < 1 + \frac{1}{\varphi} = \varphi.$$

Now, assume that $M = \emptyset$. This means that $\Pi(C)$ consists of a prefix of L (the smallest items). Let $i_1 \succ \cdots \succ i_k$ be the items in $\Pi(C) \setminus \text{OPT}(\mathcal{I}, C)$, and let $j_1 \succ \cdots \succ j_l$ be the items in $\text{OPT}(\mathcal{I}, C) \setminus \Pi(C)$. As $\Pi(C)$ consists of a prefix of L, we have $|\Pi(C)| \geq |\text{OPT}(\mathcal{I}, C)|$ and thus $k \geq l$. If $k = 0$, the claim trivially holds. Otherwise, since M is empty, we have $j_l \succ i_1$. It suffices to show $v(j_h) \leq \varphi v(i_h)$ for all $h \leq l$. To this end, we consider any fixed $h \leq l$. From $v(\{i_1, \ldots, i_{h-1}\}) \leq v(\{j_1, \ldots, j_{h-1}\})$ it follows that

$$v(j_h) \leq v(\text{OPT}(\mathcal{I}, C)) - v(\{j_1, \ldots, j_{h-1}\}) \leq C - v(\{i_1, \ldots, i_{h-1}\}).$$

This implies that j_h cannot precede all items of $\{i_h, \ldots, i_k\}$ in Π, as $j_h \notin \Pi(C)$. Hence, there is an item $i'' \in \{i_h, \ldots, i_k\}$ that precedes j_h in Π. Since $j_h \succ i''$, in the iteration when UNIVERSALUD inserted j_h into Π, i'' was already present. From the fact that i'' ended up preceding j_h it follows that j_k was compared with i'' and thus $v(j_h) < \varphi v(i'') \leq \varphi v(i_h)$. We obtain

$$\frac{v(\text{OPT}(\mathcal{I}, C))}{v(\Pi(C))} \leq \frac{v(\text{OPT}(\mathcal{I}, C) \setminus \Pi(C))}{v(\Pi(C) \setminus \text{OPT}(\mathcal{I}, C))} = \frac{\sum_{h=1}^{l} v(j_h)}{\sum_{h=1}^{k} v(i_h)} \leq \frac{\sum_{h=1}^{l} \varphi v(i_h)}{\sum_{h=1}^{l} v(i_h)} = \varphi.$$

∎

A naive implementation of UNIVERSALUD runs in time $\Theta(n^2)$. We improve this running time to $\Theta(n \log n)$. Observe that this is still best-possible in the sense of Theorem 9, since the proof only used unit densities.

THEOREM 12. *The algorithm* UNIVERSALUD *can be implemented to run in time* $\Theta(n \log n)$.

Proof. To improve the running time from the nave $\Theta(n^2)$, we maintain a balanced search tree T that stores a subset of the items in Π sorted decreasingly by their sizes. Whenever an item gets inserted to the front of Π, and only then, we also insert it into T. This way, the items in T remain sorted by their positions in Π throughout the execution of the algorithm. We need an efficient way of finding, in each iteration r of UNIVERSALUD (Algorithm 3), the first item i in $\Pi^{(r)}$ for which $v(L_r) \geq \varphi v(i)$, or detecting that no such item exists. We claim that, if such an item exists, it is stored in T and can thus be found in time $\Theta(\log n)$.

It suffices to show that for every item $i \in T$ and its predecessor j in T we have that none of the items that precede i in Π are smaller than j. To see this, we argue that none of the items between j and i in Π are smaller than j. We can then repeat the argument for j and its predecessor j', etc. For the sake of contradiction, let i' be the first item between j and i with $v(i') < v(j)$. None of the items between j and i' are smaller than j, hence both j and i' are inserted into Π earlier than all of them. Let r be the iteration in which j is inserted into Π. Since i' is inserted earlier into Π, and since j is inserted to the front of $\Pi^{(r)}$, i' is at the front of $\Pi^{(r-1)}$. This is a contradiction to i' not being in T. ∎

We now establish that UNIVERSALUD is best possible, even if we permit non-universal policies.

THEOREM 13. *There are instances of the knapsack problem with unknown capacity where no policy achieves a robustness factor of $\varphi - \delta$, for any $\delta > 0$, even when all items have unit density.*

Proof. Consider an instance of the knapsack problem with unknown capacity with five items of unit density and values equal to $v_1 = 1 + \varepsilon, v_2 = 1 + \varepsilon, v_3 = 2/\varphi, v_4 = 1 + 1/\varphi^2, v_5 = \varphi$, for sufficiently small $\varepsilon > 0$, i.e., $\varepsilon < \delta/(\varphi - \delta)$. We show that no algorithm achieves a robustness factor of $\varphi - \delta$ for this instance. To this end we consider an arbitrary algorithm $\mathcal{A}$ and distinguish different cases depending on which item the algorithm tries to pack first.

(a) If $\mathcal{A}$ tries item 1 or item 2 first, it cannot fit any additional item for a capacity equal to $v_5 = \varphi$, as even $v_1 + v_2 > \varphi$. For this capacity $\mathcal{A}$ is worse by a factor of $\varphi/(1 + \varepsilon) > \varphi - \delta$ than the optimum solution, which packs item 5.

(b) If $\mathcal{A}$ tries item 3 first, it cannot fit any additional item for a capacity equal to $v_1 + v_2 = 2 + 2\varepsilon$, as even $v_3 + v_1 > 2 + 2\varepsilon$. For this capacity $\mathcal{A}$ is worse by a factor of $(1 + \varepsilon)\varphi > \varphi - \delta$ than the optimum solution which packs items 1 and 2.

(c) If $\mathcal{A}$ tries item 4 first, it cannot fit any additional item for a capacity equal to $v_2 + v_3 = 1 + 2/\varphi + \varepsilon$, as even $v_4 + v_1 = 2 + 1/\varphi^2 + \varepsilon > 1 + 2/\varphi + \varepsilon$. For this capacity $\mathcal{A}$ is worse by a factor of $\frac{1+2/\varphi+\varepsilon}{1+1/\varphi^2} > \frac{\varphi+1/\varphi}{1+1/\varphi^2} = \varphi > \varphi - \delta$ than the optimum solution which packs items 2 and 3.

(d) If $\mathcal{A}$ tries item 5 first, it cannot fit any additional item for a capacity equal to $v_3 + v_4 = \varphi + 1$, as even $v_5 + v_1 = \varphi + 1 + \varepsilon > \varphi + 1$. For this capacity $\mathcal{A}$ is worse by a factor of $\frac{\varphi+1}{\varphi} = \varphi > \varphi - \delta$ than the optimum solution which packs items 3 and 4. ∎

5. Hardness. Although we can always find a 2-robust universal policy in polynomial time, we show in this section that, for any fixed $\alpha \in (1, \infty)$, it is intractable to decide whether a given universal policy is α-robust. This hardness result also holds for instances with unit densities when α is part of the input. As the final – and arguably the most interesting – result of this section, we establish coNP-hardness of the problem to decide for a given instance and given $\alpha > 1$, whether the instance admits a universal policy with robustness factor α. All proofs rely on the hardness of the following version of SUBSETSUM.

LEMMA 14. *Let $W = \{w_1, w_2, \ldots, w_n\}$ be a set of positive integer weights and $T \leq \sum_{k=1}^{n} w_k$ be a target sum. The problem of deciding whether there is a subset $U \subseteq W$ with $\sum_{w \in U} w = T$ is* NP*-complete, even when*

1. *$T = 2^k$ for some integer $k \geq 3$,*
2. *all weights are in the interval $[2, T/2)$,*
3. *for every weight $w \in W$ it holds that $|2^k - w| \geq 2$ for all $k \in \mathbb{N}$.*

Proof. Without Properties 1 to 3, the SUBSETSUM problem is well known to be NP-complete (e.g., Garey and Johnson [16]). Given an instance (W, T) of this classical problem, we construct an equivalent instance with Properties 1 to 3. We first multiply all weights in W as well as the target sum T with 6 to obtain an equivalent instance (W', T'). In the new instance, all weights are even but not a power of 2, hence they have distance at least 2 to the closest power of 2. We set $T'' = 2^\sigma$, with $\sigma = \lceil \log_2(T' + \sum_{w' \in W'} w') \rceil + 2$ and define two new weights

$$u = \left\lfloor \frac{T'' - T'}{2} \right\rfloor, \quad w = \left\lceil \frac{T'' - T'}{2} \right\rceil.$$

We set $W'' = W' \cup \{u, w\}$ to obtain the final instance (W'', T''). Properties 1 and 2 are satisfied by construction. Also, any solution to the instance (W'', T'') has to include both u and w, since $T'' > 4 \cdot \sum_{w' \in W'} w'$. Hence, the instance remains equivalent to the original instance (W, T). Since $T'' - T' > 3T''/4$, and since T'' is a power of two, the new items u and w are far enough from the closest power of 2 (which either is $T''/2$ or $T''/4$). ∎

We first show that it is intractable to determine the robustness factor of a given universal policy.

Theorem 15. *For any fixed and polynomially representable $\alpha > 1$ it is* coNP*-complete to decide whether a given universal policy for the knapsack problem with unknown capacity is α-robust.*

Proof. Regarding the membership in coNP, note that if a universal policy Π is not α-robust, then there is a capacity C such that $v(\Pi(C)) < v(\textsc{Opt}(\mathcal{I}, C))/\alpha$. Thus, C together with $\textsc{Opt}(\mathcal{I}, C)$ is a certificate for Π not being an α-robust solution.

For the proof of coNP-hardness, we reduce from the variant of SubsetSum specified in Lemma 14. An instance of this problem is given by a set $W = \{w_1, w_2, \dots, w_n\}$ of positive integer weights in the range $[2, T/2)$ and a target sum $T = 2^k$ for some integer $k \geq 3$. Let $\alpha > 1$ be polynomially representable. We may assume without loss of generality that $\alpha > \frac{T}{T-1}$ as we can ensure this property by multiplying T and all items in W by a sufficiently large power of 2.

We construct an instance $\mathcal{I}$ and a sequence Π such that Π is an α-robust universal policy for $\mathcal{I}$ if and only if the instance of SubsetSum given by W and T has no solution. To this end, we introduce for each weight $w \in W$ an item with value and size equal to w. In this way, the optimal knapsack solution for capacity T is at least T if the instance of SubsetSum has a solution. Furthermore, we introduce a set of additional items that make sure that the robustness factor for all capacities except T is at most α while maintaining the property that the optimal knapsack solution for capacity T is strictly less than T if the instance of SubsetSum has no solution.

We now explain the construction of $\mathcal{I}$ and Π is detail. Let $\epsilon = \frac{\alpha(T-1)-T}{\alpha(T-1)-1}$, i.e., $\alpha = \frac{T-\epsilon}{(T-1)(1-\epsilon)}$. Note that $\epsilon \in (0,1)$ by our assumptions on T and α. For each weight $w \in W$, we introduce an item i_w with $l(i_w) = v(i_w) = w$. The set of these items is called *regular* and is denoted by $\mathcal{I}_{\text{reg}}$. Furthermore, we introduce a set of auxiliary items. Let $m = \log_2 T - 1$. Then, for each $k \in \{0, 1, \dots, m\}$, we introduce an auxiliary item j_k with size $l(j_k) = 2^k$ and value $v(j_k) = 2^k\,(1-\epsilon)$. Denoting the set of auxiliary items by $\mathcal{I}_{\text{aux}}$, we have $l(\mathcal{I}_{\text{aux}}) = \sum_{k=0}^{m} 2^k = T - 1$. Finally, we introduce a dummy item d with $l(d) = T + 1$ and

$$v(d) = \frac{1-\epsilon}{\epsilon}\,(v(\mathcal{I}_{\text{aux}}) + v(\mathcal{I}_{\text{reg}})) = \frac{1-\epsilon}{\epsilon}\left((T-1)(1-\epsilon) + \sum_{w \in W} w\right).$$

The universal policy Π is defined as $\Pi = (d, j_m, j_{m-1}, \dots, j_0, i_{w_n}, i_{w_{n-1}}, \dots, i_{w_1})$. The hardness proof relies on the claim that Π is a $\frac{1}{1-\epsilon}$-robust universal policy for all capacities except T, i.e.,

$$v(\textsc{Opt}(\mathcal{I}, C)) \leq \frac{1}{1-\varepsilon} v(\Pi(C)) \text{ for all } C \neq T. \tag{2}$$

As all item sizes are integer, it suffices to consider integer capacities. To prove (2), let us first consider capacities $C \leq T - 1$. Since the density of each item with size not larger than $T - 1$ is bounded from above by 1, it is sufficient to show that

$v(\Pi(C)) = C(1-\epsilon)$. To this end, we show that every capacity $C \in \{1, \dots, 2^{m+1}-1 = T-1\}$ is packed without a gap by the exponentially decreasing sequence of items $j_m, j_{m-1}, \dots, j_0$. We prove this statement by induction over m. For $m = 0$, the statement is true, since there is only a single item with length 1, which packs the capacity $C = 1$ optimally. Now assume that the statement is true for all $m' < m$ and consider the sequence $j_m, j_{m-1}, \dots, j_0$. We distinguish two cases. For capacities $C \in \{2^m, \dots, 2^{m+1}-1\}$, item j_m is packed and, using the induction hypothesis, the residual capacity $\tilde{C} = C - 2^m \leq 2^{m+1} - 1 - 2^m \leq 2^m - 1$ can be packed without a gap by the remaining sequence $j_{m-1}, j_{m-2}, \dots, j_0$. For capacities $C < 2^m$, item j_m is not packed, and, again using the induction hypothesis, we derive that C can be packed by $j_{m-1}, \dots, j_0$. This completes the proof of our claim for $C \leq T-1$.

Let us now consider our claim for capacities $C \geq T+1$. In this case, $d \in \Pi(C)$ and we can trivially bound the robustness factor of Π by observing that

$$\frac{v(\textsc{Opt}(\mathcal{I}, C))}{v(\Pi(C))} \leq \frac{v(\mathcal{I})}{v(d)} = 1 + \frac{(T-1)(1-\epsilon) + \sum_{w \in W} w}{v(d)} = 1 + \frac{\epsilon}{1-\epsilon} = \frac{1}{1-\epsilon}.$$

We proceed to show that Π is an α-robust universal policy if and only if the instance of SubsetSum given by W and T has no solution. Let us first assume that the instance of SubsetSum has no solution. We prove that Π is α-robust. For all capacities except T this is clear from claim (2). For capacity T, we argue as follows: As there is no packing of T with items of density 1, we bound $v(\textsc{Opt}(\mathcal{I}, T))$ from above by $(T-1) + (1-\epsilon)$, whereas Π packs all auxiliary items. We get

$$\frac{v(\textsc{Opt}(\mathcal{I}, T))}{v(\Pi(T))} \leq \frac{(T-1) + (1-\epsilon)}{(T-1)(1-\epsilon)} = \alpha.$$

Now, assume that the instance of SubsetSum has a solution. Then, $v(\textsc{Opt}(T)) = T$ and thus

$$\frac{v(\textsc{Opt}(\mathcal{I}, T))}{v(\Pi(T))} = \frac{T}{(T-1)(1-\epsilon)} > \alpha,$$

and we conclude that Π is not α-robust. ∎

We give a result similar to Theorem 15 for unit densities. Note that this time we require α to be part of the input.

Theorem 16. *It is* coNP*-complete to decide whether, for given $\alpha > 1$, a given universal policy for the oblivious knapsack problem is α-robust, even when all items have unit density.*

Proof. Membership in coNP follows from Theorem 15. To prove hardness, we again reduce from SubsetSum (Lemma 14) using a similar construction as in the proof of Theorem 15. Let the set $W = \{w_1, \dots, w_n\}$ of weights and the target sum $T \geq 8$ of an instance of SubsetSum be given, with $w_1 \leq w_2 \leq \dots \leq w_n$. We proceed to explain the construction of a universal policy Π for which the decision whether Π is α-robust is coNP-hard, for some $\alpha > 1$.

For each weight $w \in W$, we introduce an item i_w with value $v(i_w) = w$. The set of these items is called *regular* and is denoted by $\mathcal{I}_{\text{reg}}$. Let $m = \log_2 T - 1$ and $\epsilon = 1/T^2$. For each $k \in \{0, \dots, m\}$, we introduce an auxiliary item j_k with value $v(j_k) = 2^k(1-\epsilon)$. Denoting the set of auxiliary items by $\mathcal{I}_{\text{aux}}$, we have $v(\mathcal{I}_{\text{aux}}) = (1-\epsilon)\sum_{k=0}^{m} 2^k = (1-\epsilon)(T-1)$. We further introduce a set of dummy items $\mathcal{I}_{\text{dum}} = \{d_0, \dots, d_{m'}\}$, where

$m' = \lceil \log_2 w_n \rceil$. We set $v(d_k) = T \cdot 2^k$ for each $k \in \{1, \ldots, m'\}$, and $v(d_0) = T + \varepsilon$. The values of the dummy items sum up to $v(\mathcal{I}_{\text{dum}}) = (T+\epsilon) + T \sum_{k=1}^{m'} 2^k = T(2^{m'+1} - 1) + \epsilon$. In total, the sum of the values of all dummy and auxiliary items is

$$S = v(\mathcal{I}_{\text{aux}}) + v(\mathcal{I}_{\text{dum}}) = (1-\epsilon)(T-1) + T(2^{m'+1} - 1) + \epsilon. \tag{3}$$

Finally, we define the sequence Π as

$$\Pi = (d_{m'}, d_{m'-1}, \ldots, d_0, j_m, j_{m-1}, \ldots, j_0, i_{w_n}, i_{w_{n-1}}, \ldots, i_{w_1}),$$

i.e., Π first tries to pack the dummy items in decreasing order, then the auxiliary items in decreasing order, and finally the regular items in non-increasing order. Let $\alpha = \frac{T-\epsilon}{(1-\epsilon)(T-1)}$. We proceed to prove the statement of the theorem by showing that Π is an α-robust universal policy if and only if the instance (W, T) of SubsetSum has no solution. To this end, we first prove that Π is always an α-robust universal policy for all capacities except the *critical* capacities in the interval $\big[T - \epsilon T, T + \epsilon\big)$. Then, we argue that Π is α-robust for the critical capacities if and only if the instance (W, T) of SubsetSum has no solution.

We start by proving that $v(\Pi(C))$ is within an α-fraction of $v(\text{Opt}(C))$ for all capacities $C \in [0, T - \epsilon T)$. Since the regular items are of integer values and the values of the auxiliary items each are an $(1-\epsilon)$-fraction of an integer, only capacities C for which the ratio $C/\lceil C \rceil$ is not smaller than $1-\epsilon$ can be packed without a gap. Otherwise, the value of an optimal solution is bounded from above by $\lfloor C \rfloor$. For capacities $C \in [0, T - \epsilon T)$, we obtain

$$v(\text{Opt}(\mathcal{I}, C)) \leq \begin{cases} C, & \text{if } C/\lceil C \rceil \geq 1 - \epsilon \\ \lfloor C \rfloor, & \text{otherwise.} \end{cases} \tag{4}$$

The value packed by Π is given by

$$v(\Pi(C)) = \begin{cases} (1-\epsilon)\lceil C \rceil, & \text{if } C/\lceil C \rceil \geq 1 - \epsilon \\ (1-\epsilon)\lfloor C \rfloor, & \text{otherwise.} \end{cases} \tag{5}$$

From (4) and (5) it follows that

$$v(\text{Opt}(\mathcal{I}, C)) \leq \frac{1}{1-\epsilon} v(\Pi(C)) < \alpha\, v(\Pi(C)) \tag{6}$$

for all $C \in [0, T - \epsilon T)$.

Next, we prove that Π is within an α-fraction of an optimal solution for all capacities $C \in [T + \epsilon, S]$. We distinguish two cases for each such capacity C.

First case: $\mathcal{I}_{\text{aux}} \subset \Pi(C)$, i.e., all auxiliary items are packed by Π. Since, in Π, the dummy item d_0 with value $T + \epsilon$ precedes all auxiliary items, and since $C \geq T + \epsilon$, this case can only occur for capacities

$$C \geq v(d_0) + v(\mathcal{I}_{\text{aux}}) = T + \varepsilon + (1-\varepsilon)(T-1) = 2(T+\varepsilon) - (1 + \varepsilon T). \tag{7}$$

On the other hand, the gap $C - v(\Pi(C))$ is at most the gap left after trying all dummy items and packing all auxiliary items, i.e., $C - v(\Pi(C)) < v(d_0) - v(\mathcal{I}_{\text{aux}}) = T + \epsilon - (1-\epsilon)(T-1) = 1 + \epsilon T$. Thus,

$$\begin{aligned} \frac{v(\text{Opt}(\mathcal{I}, C))}{v(\Pi(C))} < \frac{C}{C - (1+\epsilon T)} &\overset{(7)}{\leq} \frac{2(T+\varepsilon) - (1+\varepsilon T)}{2(T+\varepsilon) - 2(1+\varepsilon T)} \\ &= \frac{(T+\epsilon) - (1+\epsilon T)/2}{(T+\varepsilon) - (1+\varepsilon T)} \overset{T \geq 8}{<} \frac{T-\epsilon}{(1-\epsilon)(T-1)} = \alpha. \end{aligned}$$

Second case: $\mathcal{I}_{aux} \setminus \Pi(C) \neq \emptyset$, i.e., not all auxiliary items are packed. This implies that the gap $C - v(\Pi(C))$ is at most $1 - \epsilon$. We calculate

$$\frac{v(\textsc{Opt}(\mathcal{I}, C))}{v(\Pi(C))} < \frac{C}{C - (1-\epsilon)} \overset{C \geq T+\varepsilon}{\leq} \frac{T+\epsilon}{T + 2\epsilon - 1} \overset{\varepsilon = 1/T^2}{<} \frac{T-\epsilon}{(1-\epsilon)(T-1)} = \alpha.$$

Next, we consider capacities $C \in (S, v(\mathcal{I}_{\text{aux}} \cup \mathcal{I}_{\text{dum}} \cup \mathcal{I}_{\text{reg}})]$. For these capacities, all dummy items and all auxiliary items are packed by Π. Using that the gap $C - \Pi(C)$ is at most w_n, we obtain

$$\begin{aligned} \frac{v(\textsc{Opt}(\mathcal{I}, C))}{v(\Pi(C))} &\leq \frac{C}{C - w_n} \overset{C>S}{<} \frac{S}{S - w_n} \overset{S > T2^{m'}}{<} \frac{T2^{m'}}{T2^{m'} - w_n} \\ &\leq \frac{Tw_n}{Tw_n - w_n} = \frac{T}{T-1} = \frac{T(1-\varepsilon)}{(1-\epsilon)(T-1)} < \frac{T-\epsilon}{(1-\epsilon)(T-1)} = \alpha. \end{aligned}$$

To finish the proof, let us finally consider the critical capacities $C \in \big[T - T\epsilon, T + \epsilon\big)$. We proceed to show that $v(\Pi(C))$ is within an α-fraction of $v(\textsc{Opt}(C))$ for all $C \in \big[T - T\epsilon, T + \epsilon\big)$ if and only if (W, T) does not have a solution. Let us first assume that (W, T) does not have a solution. Then, $v(\textsc{Opt}(C)) \leq T - \epsilon$ and we obtain

$$\frac{v(\textsc{Opt}(\mathcal{I}, C))}{v(\Pi(C))} \leq \frac{T - \epsilon}{(T-1)(1-\epsilon)} = \alpha,$$

for all $C \in \big[T - T\epsilon, T + \epsilon\big)$. If, on the other hand, (W, T) has a solution, then $v(\textsc{Opt}(T)) = T$, implying that

$$\frac{v(\textsc{Opt}(\mathcal{I}, T))}{v(\Pi(T))} = \frac{T}{(T-1)(1-\epsilon)} > \alpha,$$

i.e., Π is not an α-robust universal policy. ∎

Finally, we prove that it is hard to decide whether a given instance admits an α-robust universal policy when α is part of the input.

THEOREM 17. *It is* coNP*-hard to decide whether, for given $\alpha > 1$, an instance of the knapsack problem with unknown capacity admits an α-robust universal policy, even when all items have unit density.*

Proof. We again reduce from SUBSETSUM. To this end, let (W, T) be an instance of SUBSETSUM (Lemma 14), let $\mathcal{I}$ be the set of items constructed from (W, T) in the proof of Theorem 16, and let $\alpha = \frac{T-\epsilon}{(1-\epsilon)(T-1)}$. We proceed to show that $\mathcal{I}$ admits an α-robust universal policy if and only if the instance (W, T) of SUBSETSUM has no solution.

For the case that (W, T) has no solution, an α-robust universal policy is constructed in the proof of Theorem 16. Thus, it suffices to show that if (W, T) has a solution, $\mathcal{I}$ does not admit an α-robust universal policy.

First, we claim that any α-robust universal policy Π contains the auxiliary items in decreasing order. Otherwise, for the sake of contradiction, let j be the first auxiliary item in Π that is preceded by a smaller auxiliary item i. Consider the capacity $C = v(j)$. As all dummy items are larger than $T > C$, only auxiliary and regular items can be in $\Pi(C)$. Since i precedes j, we have $j \notin \Pi(C)$.

If $\Pi(C)$ contains only auxiliary items, since the sum of the values of the auxiliary items smaller than $v(j)$ is $v(j) - (1 - \epsilon)$, we can use that $j \notin \Pi(C)$ to obtain $v(\Pi(C)) \leq$

$v(j) - (1-\epsilon) < \lfloor v(j) \rfloor$. If $\Pi(C)$ contains a regular item i', then $\frac{C - v(i')}{\lceil C - v(i') \rceil} < 1 - \varepsilon$, and hence the gap $C - v(i')$ cannot be packed with a value more than $\lfloor C - v(i') \rfloor$. It follows that $v(\Pi(C)) \leq \lfloor v(j) \rfloor$. In either case we have

$$\frac{v(\mathrm{OPT}(\mathcal{I}, C))}{v(\Pi(C))} \geq \frac{v(j)}{\lfloor v(j) \rfloor} \overset{v(j) \leq (1-\varepsilon)T/2}{\geq} \frac{(1-\varepsilon)T/2}{\lfloor (1-\varepsilon)T/2 \rfloor}$$
$$= \frac{(1-\varepsilon)T/2}{T/2 - 1} \overset{\varepsilon = 1/T^2}{>} \frac{T - \epsilon}{(T-1)(1-\epsilon)} = \alpha.$$

This is a contradiction to the assumption that Π is α-robust. We conclude that the auxiliary items appear in Π in decreasing order.

Second, we claim that if $\Pi(T)$ contains a regular item, then Π is not α-robust. By the argument above, we may assume that the auxiliary items in Π are ordered decreasingly. Let i be the regular item contained in $\Pi(T)$ that appears first in Π. Consider the capacity $C = (v(i) + 1)(1 - \epsilon)$. The auxiliary items that appear before i in Π (if any) are ordered decreasingly. All of them must be larger than $v(i)$, otherwise, the gap left after packing them for capacity T would be too small to fit i. By Lemma 14, we have that neither $v(i)$ nor $v(i)+1$ are a power of 2, thus $\Pi(C)$ does not contain any of the auxiliary items preceding i. All regular items that appear before i in Π are larger than $v(i)$, since they are not in $\Pi(T)$. Hence, $\Pi(C)$ does not contain any regular items except i. We conclude that $\Pi(C) = \{i\}$. On the other hand, C is an integer multiple of $1 - \epsilon$ and can be packed without a gap by auxiliary items only. We obtain

$$\frac{v(\mathrm{OPT}(C))}{v(\Pi(C))} = \frac{C}{v(i)} = \frac{(v(i)+1)(1-\varepsilon)}{v(i)} \overset{v(i) \leq T/2}{\geq} \frac{(T/2+1)(1-\varepsilon)}{T/2} \overset{\varepsilon = 1/T^2}{>} \alpha.$$

We conclude that if an α-robust universal policy Π exists, then $\Pi(T)$ does not contain regular items. It follows that $\Pi(T) = \mathcal{I}_{\mathrm{aux}}$ and, thus, $v(\Pi(T)) = (T-1)(1-\epsilon)$. Using that the SUBSETSUM instance (W, T) has a solution, we obtain

$$\frac{v(\mathrm{OPT}(\mathcal{I}, T))}{v(\Pi(T))} \geq \frac{T}{(T-1)(1-\epsilon)} > \alpha, \qquad \square$$

which implies that no α-robust universal policy exists.

6. Final remarks. In this work, we presented universal sequencing algorithms for the knapsack problem with unknown capacity in which non-fitting items can be discarded. Our deterministic algorithms construct solutions which achieve best-possible robustness factors. Surprisingly, best-possible robustness factors can already be obtained by universal policies, i.e., policies that attempt to fix the items in a universal, non-adaptive order. We showed how such orders can be computed in $\mathcal{O}(n \log n)$.

It remains an interesting open question how much the robustness factors could be improved when allowing randomized strategies. Randomized universal sequences have been derived recently in the context of scheduling [14], matching [32], cardinality-constrained knapsack [29] and more general independence systems [32, 29]. Our algorithms do not seem to directly suggest a natural randomized procedure.

Finally, we point out an interesting interpretation of the capacity-oblivious knapsack models with and without discarding items by using feasibility oracles. The knapsack model without discarding items [21, 33] adds items until the first item does not fit anymore, whereas in our model the packing would proceed after discarding the not-fitting item. The latter behaviour can be modelled by considering the model without discarding items and giving access to a certain weak feasibility oracle. For a given item, the feasibility oracle either returns the information that the item does not fit in the knapsack, or it irrevocably packs the item if it fits. Our results transfer directly to such a model. Along these lines one may ask for the gain when an algorithm is granted access to an even stronger oracle that receives as input an item and returns the information whether this item fits into the knapsack—without enforcing to pack the item. It is straightforward to verify that our lower bounds in Theorems 10 and 13 are still valid in this case. Thus, our algorithms are optimal even though they utilize only a weak oracle. The case of even more powerful oracles that answer queries for item sets is left for future research.

REFERENCES

[1] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg, *A knapsack secretary problem with applications*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 10th International Workshop, APPROX 2007, and 11th International Workshop, RANDOM 2007, Princeton, NJ, USA, August 20-22, 2007, Proceedings, M. Charikar, K. Jansen, O. Reingold, and J. D. P. Rolim, eds., vol. 4627 of Lecture Notes in Computer Science, Springer, 2007, pp. 16–28.

[2] M. A. Bender, R. Cole, and E. D. Demaine, *Scanning and traversing: maintaining data for traversals in a memory hierarchy*, in Proceedings of the 10th European Symposium on Algorithms (ESA), 2002, pp. 139–151.

[3] D. Bertsimas and M. Sim, *Robust discrete optimization and network flows*, Mathematical Programming, 98 (2003), pp. 49–71.

[4] A. Bhalgat, A. Goel, and S. Khanna, *Improved approximation results for stochastic knapsack problems*, in Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2011, pp. 1647–1665.

[5] H. Böckenhauer, D. Komm, R. Královic, and P. Rossmanith, *The online knapsack problem: Advice and randomization*, Theor. Comput. Sci., 527 (2014).

[6] P. C. Bouman, J. M. van den Akker, and J. A. Hoogeveen, *Recoverable robustness by column generation*, in Proceedings of the 19th European Symposium on Algorithms (ESA), 2011, pp. 215–226.

[7] C. Büsing, A. M. Koster, and M. Kutschka, *Recoverable robust knapsacks: the discrete scenario case*, Optimization Letters, 5 (2011), pp. 379–392.

[8] J. L. Carter and M. N. Wegman, *Universal classes of hash functions*, Journal of Computer and System Sciences, 18 (1979), pp. 143–154.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, 3 ed., 2009.

[10] B. C. Dean, M. X. Goemans, and J. Vondrák, *Approximating the stochastic knapsack problem: The benefit of adaptivity*, Math. Oper. Res., 33 (2008), pp. 945–964.

[11] V. G. Deineko, R. Rudolf, and G. J. Woeginger, *Sometimes travelling is easy: The master tour problem*, in Proceedings of the 3rd European Symposium on Algorithms (ESA), 1995, pp. 128–141.

[12] D. Diodati, A. Navarra, and C. M. Pinotti, *Online knapsack of unknown capacity:*, in Proceedings of the 14th International Symposium on Experimental Algorithms (SEA), 2015, pp. 165–177.

[13] Y. Disser, N. Megow, M. Klimm, and S. Stiller, *Packing a knapsack of unknown capacity*, in Proceedings of the 31st Symposium on Theoretical Aspects of Computer Science (STACS), 2014, pp. 276–287.

[14] L. Epstein, A. Levin, A. Marchetti-Spaccamela, N. Megow, J. Mestre, M. Skutella, and L. Stougie, *Universal sequencing on an unreliable machine*, SIAM Journal on Computing, 41 (2012), pp. 565–586.

[15] M. Frigo, C. Leiserson, H. Prokop, and S. Ramachandran, *Cache-oblivious algorithms*, in Proceedings of the 40th Symposium on Foundations of Computer Science (FOCS), 1999, pp. 285–297.

[16] M. R. Garey and D. S. Johnson, *Computers and Intractability, A Guide to the Theory of*

NP-Completeness, W.H. Freeman and Company, 1979.

[17] K.-S. Goetzmann, S. Stiller, and C. Telha, *Optimization over integers with robustness in cost and few constraints*, in Proceedings of the 9th Workshop on Approximation and Online Algorithms (WAOA), 2011, pp. 89–101.

[18] X. Han, Y. Kawase, and K. Makino, *Randomized algorithms for online knapsack problems*, Theoretical Computer Science, 562 (2015), pp. 395–405.

[19] X. Han, Y. Kawase, K. Makino, and H. Guo, *Online removable knapsack problem under convex function*, Theor. Comput. Sci., 540 (2014), pp. 62–69.

[20] X. Han and K. Makino, *Online removable knapsack with limited cuts*, Theor. Comput. Sci., 411 (2010), pp. 3956–3964.

[21] J. Hartline and A. Sharp, *An incremental model for combinatorial maximization problems*, in Proceedings of the 5th International Conference on Experimental Algorithms (WEA), 2006, pp. 36–48.

[22] R. Hassin and S. Rubinstein, *Robust matchings*, SIAM Journal on Discrete Mathematics, 15 (2002), pp. 530–537.

[23] K. Iwama and S. Taketomi, *Removable online knapsack problems*, in Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP), P. Widmayer, F. T. Ruiz, R. M. Bueno, M. Hennessy, S. Eidenbenz, and R. Conejo, eds., vol. 2380 of Lecture Notes in Computer Science, Springer, 2002, pp. 293–305.

[24] K. Iwama and G. Zhang, *Online knapsack with resource augmentation*, Inf. Process. Lett., 110 (2010), pp. 1016–1020.

[25] L. Jia, G. Lin, G. Noubir, R. Rajaraman, and R. Sundaram, *Universal approximations for TSP, Steiner tree, and set cover*, in Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC), 2005, pp. 386–395.

[26] N. Kakimura, K. Makino, and K. Seimi, *Computing knapsack solutions with cardinality robustness*, in Proceedings of the 22nd International Conference on Algorithms and Computation (ISAAC), 2011, pp. 693–702.

[27] A. J. Kleywegt and J. D. Papastavrou, *The dynamic and stochastic knapsack problem*, Operations Research, 46 (1998), pp. 17–35.

[28] A. J. Kleywegt and J. D. Papastavrou, *The dynamic and stochastic knapsack problem with random sized items*, Operations Research, 49 (2001), pp. 26–41.

[29] Y. Kobayashi and K. Takazawa, *Randomized strategies for cardinality robustness in the knapsack problem*, in Proceedings of the Thirteenth Workshop on Analytic Algorithmics and Combinatorics (ANALCO), 2016, pp. 25–33.

[30] B. Korte and J. Vygen, *Combinatorial Optimization. Theory and Algorithms.*, Springer, 2nd ed., 2002.

[31] A. Marchetti-Spaccamela and C. Vercellis, *Stochastic on-line knapsack problems*, Math. Program., 68 (1995), pp. 73–104.

[32] J. Matuschke, M. Skutella, and J. A. Soto, *Robust randomized matchings*, in Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2015, pp. 1904–1915.

[33] N. Megow and J. Mestre, *Instance-sensitive robustness guarantees for sequencing with unknown packing and covering constraints*, in Proceedings of the 4th Conference on Innovations in Theoretical Computer Science (ITCS), 2013, pp. 495–504.

[34] R. Merkle and M. E. Hellman, *Hiding information and signatures in trapdoor knapsacks*, IEEE Transactions on Information Theory, 24 (1978), pp. 525–530.

[35] M. Monaci and U. Pferschy, *On the robust knapsack problem*, in Proceedings of the 10th Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW), 2011, pp. 207–210.

[36] J. Noga and V. Sarbua, *An online partially fractional knapsack problem*, in Proceedings of the 8th International Symposium on Parallel Architectures, Algorithms, and Networks (ISPAN), IEEE Computer Society, 2005, pp. 108–112.

[37] C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley, 1994.

[38] H. Räcke, *Survey on oblivious routing strategies*, in Proceedings of the 5th Conference on Computability in Europe: Mathematical Theory and Computational Practice (CiE), 2009, pp. 419–429.

[39] A. Shamir, *A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem*, in Proceedings of the 23rd Symposium on Foundations of Computer Science (FOCS), 1982, pp. 145–152.

[40] R. V. Slyke and Y. Young, *Finite horizon stochastic knapsacks with applications to yield management*, Operations Research, 48 (2000), pp. 155–172.

[41] C. Thielen, M. Tiedemann, and S. Westphal, *The online knapsack problem with incremental*

capacity. Mathematical Methods of Operations Research, to appear, 2016.
[42] L. G. Valiant and G. J. Brebner, *Universal schemes for parallel communication*, in Proceedings of the 13th Annual ACM Symposium on Theory of Computing (STOC), 1981, pp. 263–277.
[43] G. Yu, *On the max-min 0-1 knapsack problem with robust optimization applications*, Operations Research, 44 (1996), pp. 407–415.