On the Advice Complexity of the Knapsack Problem

Report

Author(s):
Böckenhauer, Hans-Joachim; Komm, Dennis; Královič, Richard; Rossmanith, Peter

Publication date:
2011

Permanent link:
https://doi.org/10.3929/ethz-a-007313682

Originally published in:
Technical report 740

On the Advice Complexity of the Knapsack Problem*

Hans-Joachim Böckenhauer[1], Dennis Komm[1],
Richard Královič[1], and Peter Rossmanith[2]

[1] Department of Computer Science, ETH Zurich, Switzerland,
{hjb,dennis.komm,richard.kralovic}@inf.ethz.ch
[2] Department of Computer Science, RWTH Aachen, Germany,
rossmani@cs.rwth-aachen.de

Abstract. We study the advice complexity and the random bit complexity of the online knapsack problem: Given a knapsack of unit capacity, and n items that arrive in successive time steps, an online algorithm has to decide for every item whether it gets packed into the knapsack or not. The goal is to put as much valuable items as possible into it without exceeding its capacity.
In the model of advice complexity of online problems, one asks how many bits of advice about the unknown parts of the input are both necessary and sufficient to achieve a specific competitive ratio.
It is well-known that even the unweighted online knapsack problem does not admit any competitive deterministic online algorithm. We show that a single bit of advice helps a deterministic algorithm to become 2-competitive, but that $\Omega(\log n)$ advice bits are necessary to further improve the deterministic competitive ratio. This is the first time that such a phase transition for the number of advice bits has been observed for any problem. We also show that, surprisingly, instead of an advice bit, a single random bit allows for a competitive ratio of 2, and any further amount of randomness does not improve this. Moreover, we prove that, in a resource augmentation model, i. e., when allowing a little overpacking of the knapsack, a constant number of advice bits suffices to achieve a near-optimal competitive ratio. We also study the weighted version of the problem proving that, with $\mathcal{O}(\log n)$ bits of advice, we can get arbitrarily close to an optimal solution and, using asymptotically fewer bits, we are not competitive.

1 Introduction

Online problems are an important class of computing problems where the input is not known to the algorithm in advance, but is revealed stepwise, and where, in each step, a piece of output has to be produced irrevocably. The standard way to analyze the quality of an online algorithm is via the so-called *competitive analysis*. Here, the quality of the solution as produced by the online algorithm is compared to the quality of an offline algorithm that knows the complete input in advance. An introduction to the theory and applications of competitive analysis can be found in [3].

Comparing an algorithm having no knowledge about the forthcoming parts of the input with an algorithm having full knowledge of the future might only give a rough estimate of the real quality of an algorithm facing an online situation. To enable a more fine-grained analysis of the complexity of online problems, the *advice complexity* of online problems has been recently introduced [2, 5, 6]. The idea behind this concept is to measure the amount of information about the forthcoming parts of the input an online algorithm needs to be optimal or to achieve a certain competitive ratio. More precisely, in this model, the online algorithm has access to some tape containing advice bits produced by an oracle knowing the complete input, and its advice complexity is the number of bits it reads from this advice tape, i. e., the amount of information about the yet unknown input parts it needs to know for its computation. For a detailed introduction to the advice complexity of online problems, see [2, 8]. More results on the advice complexity of specific problems can be found

* This work was partially supported by ETH grant TH 18 07-3.

in [1, 6, 13], the relationship between advice complexity and randomized algorithms is discussed in [1, 12].

In this paper, we deal with an online version of the well-known knapsack problem. Here, an input consists of a set of items with specified weights and values, and a knapsack capacity. The goal is to choose a set of items with maximum value such that their total sum does not exceed the knapsack's capacity. The knapsack problem is a very well-studied hard optimization problem, for an introduction, see [7, 11]. In the online version of the knapsack problem, the items arrive one by one and the algorithm has to decide for each item whether it will pack it into the knapsack or not. These decisions may not be withdrawn at a later stage, i. e., no items can be removed from the knapsack again. It is easy to see that no deterministic online algorithm can achieve any bounded competitive ratio [14]. Thus, the existing literature on the online knapsack problem mainly considers restricted variants of the problem [16] or an average-case analysis of randomized algorithms [14].

We prove the following results in this paper: As already mentioned, it is not possible to achieve any competitive ratio with a deterministic algorithm without advice. For the unweighted version of the problem, we prove that, with a single advice bit, a competitive ratio of 2 is achievable. Moreover, for an instance of n items, any number $2 < k < \log(n-1)$ of advice bits cannot improve the competitive ratio. But, for every constant $\varepsilon > 0$, a competitive ratio of $1 + \varepsilon$ is achievable using $\mathcal{O}(\log n)$ advice bits. For computing an optimal solution, a linear number of advice bits is necessary. At first glance, these results fit well into the picture as given by the advice complexity results for other problems like paging, job shop scheduling, or disjoint path allocation [2]: Linear advice is needed for optimality, logarithmic advice for beating the best randomized algorithm, and very few bits suffice to beat a deterministic algorithm. But having a closer look, one sees that the situation is pretty much different for the knapsack problem compared to the other above-mentioned problems: This problem is the first one for which a sharp phase transition in the number of advice bits can be shown in the following sense. Even $\log n - 2$ advice bits are exactly as helpful as one bit, but $\mathcal{O}(\log n)$ bits already allow for an almost optimal solution.

A second line of research in this paper considers the random bit complexity of randomized online algorithms (without advice) for the knapsack problem. Here, it turns out that, surprisingly, a single random bit is as powerful as an advice bit, i. e., a single random bit can be used to achieve an expected competitive ratio of 2. Moreover, we prove that an arbitrary amount of additional randomness does not help at all, no randomized algorithm can achieve an expected competitive ratio better than $2 - \varepsilon$, for any $\varepsilon > 0$.

We analyze the behaviour of online algorithms with advice that are allowed to overpack the knapsack by some small constant amount of δ. In contrast to the original model, we show that, in this case, a constant number of advice bits is already sufficient to achieve a near-optimal competitive ratio.

In the last part of the paper, we study the general version of the problem. Obviously, all lower bounds carry over immediately from the unweighted problem. We show that, with less than $\log n$ advice bits, no online algorithm is competitive and that we can be arbitrarily close to the optimum when using $\mathcal{O}(\log n)$ advice bits.

The paper is organized as follows. In Section 2, we introduce our notations and present some observations on deterministic online algorithms for the online knapsack problem. Section 3.1 is devoted to the analysis of deterministic online algorithms with advice. In Section 3.2, we investigate the random bit complexity of the online knapsack problem, and in Section 3.3, we deal with the advice complexity of a resource augmentation variant of the problem. In Section 4, we consider the general, weighted knapsack problem. We conclude the paper with some remarks on future work in Section 5.

2 Preliminaries

In this section, we formally define the notions used in the following. All logarithms in this paper are taken to be binary, unless stated otherwise.

Definition 1 (Online Maximization Problem). *An online maximization problem consists of a set $\mathcal{I}$ of inputs and a cost function. Every input $I \in \mathcal{I}$ is a sequence of* requests $I = (x_1, \ldots, x_n)$. *Furthermore, a set of feasible outputs (or solutions) is associated with every I; every output is a sequence of* answers $O = (y_1, \ldots, y_n)$. *The cost function assigns a positive real value* $\mathrm{cost}(I, O)$ *to every input I and any feasible output O. If the input is clear from the context, we omit I and denote the cost of O as* $\mathrm{cost}(O)$. *For every input I, we call any output O that is feasible for I and has largest possible cost an* optimal solution of I, *denoted by* $\mathtt{Opt}(I)$.

We now formally define online algorithms with advice for online maximization problems, and their competitive ratios.

Definition 2 (Online Algorithm with Advice). *Consider an input I of an online maximization problem. An* online algorithm $\mathtt{A}$ with advice *computes the output sequence $\mathcal{A}^\phi = \mathtt{A}^\phi(I) = (y_1, \ldots, y_n)$ such that y_i is computed from $\phi, x_1, \ldots, x_i$, where ϕ is the content of the advice tape, i. e., an infinite binary sequence. We denote the costs of the computed output by* $\mathrm{cost}(\mathtt{A}^\phi(I))$. *The algorithm* $\mathtt{A}$ *is c-*competitive with advice complexity $s(n)$ *if there exists a constant α such that, for every n and for each I of length at most n, there exists some ϕ such that* $\mathrm{cost}(\mathtt{A}^\phi(I)) \geq \frac{1}{c} \cdot \mathrm{cost}(\mathtt{Opt}(I)) - \alpha$ *and at most the first $s(n)$ bits of ϕ have been accessed during the computation of $\mathtt{A}^\phi(I)$. If $\mathtt{A}$ is c-competitive for $\alpha = 0$, we call it* strictly c-competitive.

A detailed introduction into the theory of advice complexity can be found in [8].

Definition 3 (Online Knapsack Problem). *The* online knapsack problem, Knapsack *for short, is the following maximization problem. The input consists of a sequence of n items that are tuples of weights and values, i. e., $S = \{s_1, \ldots, s_n\}$, $s_i = (w_i, v_i)$, where $0 < w_i \leq 1$ and $v_i > 0$ for $i \in \{1, \ldots, n\}$. A feasible solution is any set of indices $S' \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S'} w_i \leq 1$; the goal is to maximize $\sum_{i \in S'} v_i$. The items are given an online fashion. For each item, an online algorithm $\mathtt{A}$ must specify whether this item is part of the solution or not as soon as it is offered.*

In the simple *version of* Knapsack, *denoted by* SimpleKnapsack, *each item has a value smaller than* 1 *that is equal to its weight.*

Since the value of an optimal solution for any instance of SimpleKnapsack is bounded by the constant capacity 1 of the knapsack, we only consider strict competitiveness in this paper. For simplicity, we subsequently abbreviate the term "strictly competitive" by "competitive".

3 The Unweighted Case

Let us first look at purely deterministic online algorithms.

Theorem 1 (Marchetti-Spaccamela and Vercellis [14]). *No deterministic online algorithm for* SimpleKnapsack *(and thus* Knapsack*) without advice is competitive.* □

Let us now consider an online algorithm $\mathtt{G}$ that realizes a straightforward greedy approach. This means that $\mathtt{G}$ takes any item while there is space left for it in the knapsack. Of course, this strategy also fails in general (as the last theorem implies), but for a subset of the instances it works quite well as the following observation states.

Observation 1 *Let I denote any instance of* SimpleKnapsack *where every item has a weight of $\leq \beta$. Then $\mathtt{G}$ achieves a gain of at least $1 - \beta$ or it is optimal.*

Indeed, if the sum of all weights is less than one, $\mathtt{G}$ is optimal. However, if this is not the case, the space that is not covered by $\mathtt{A}$ cannot be bigger than β.

3.1 Online Algorithms with Advice

To enable online algorithms to achieve better results, we now equip these algorithms with an advice tape as in Definition 2. At first, we study the *information content* of the problem, i. e., the number of advice bits both sufficient and necessary to produce optimal output. Obviously, there is a linear upper bound.

Theorem 2. *There exists an optimal online algorithm* `A` *for* SimpleKnapsack *using n bits of advice.*

Proof. For each of the n items, one bit of advice tells the algorithm whether this item is part of an arbitrary, but fixed, optimal solution or not. □

It might surprise that this bound is indeed tight as the next theorem shows.

Theorem 3. *Any online algorithm with advice for* SimpleKnapsack *needs at least $n-1$ bits to be optimal.*

Proof. For any n, consider the input $1/2, 1/4, \ldots, 1/2^{n-1}, s$, where the item s is defined as

$$s = 1 - \sum_{i=1}^{n-1} b_i 2^{-i},$$

for some vector $b \in \{0,1\}^{n-1}$. Consider the first $n-1$ items of the input. Any two different subsets of these items have a different sum. From this, it directly follows that, for any distinct value of b, there exists a *unique* optimal solution with gain 1. In other words: If s is "revealed" there was one "correct" choice for the algorithm.

If any online algorithm uses strictly less than $n-1$ bits, it cannot (by the pigeonhole principle) distinguish between all 2^{n-1} different inputs. Hence, it will output the same subset of the first $n-1$ items for two different input instances and thereby produces a sub-optimal solution for at least one of them. □

Next, let `A` be an online algorithm reading one bit of advice. This bit indicates whether there exists an item s within the input that has size $> 1/2$. If this bit is zero, `A` acts greedily, if it is one, `A` takes nothing until an item of size $> 1/2$ appears (and anything else afterwards).

Theorem 4. *The online algorithm* `A` *for* SimpleKnapsack *is 2-competitive.*

Proof. Suppose, there is no item with size $> 1/2$. In this case, the claim directly follows from Observation 1. However, if there exists an item of size $> 1/2$, the proof of the claim is trivial. □

This result seems counterintuitive. With merely one bit of advice and a straightforward approach we jump from an unbounded output quality to 2-competitiveness. However, any further increase of the number of advice bits does not help until a logarithmic number is reached. The above algorithm of Theorem 4 is therefore the best we can hope for when dealing with any constant number of advice bits.

Theorem 5. *Let $b < \lfloor \log(n-1) \rfloor$ and let $\varepsilon > 0$. No online algorithm for* SimpleKnapsack *using b bits of advice is better than $(2-\varepsilon)$-competitive.*

Proof. Let $\delta = \varepsilon/(4-2\varepsilon)$ and let `A` read b advice bits. Consider the class $\mathcal{I}$ of inputs I_j, for $1 \le j \le n-1$, of the form

$$\frac{1}{2}+\delta, \frac{1}{2}+\delta^2, \ldots, \frac{1}{2}+\delta^j, \frac{1}{2}-\delta^j, \frac{1}{2}+\delta, \ldots, \frac{1}{2}+\delta,$$

where the item $\frac{1}{2}+\delta$ appears $n-j-1$ times at the end of the instance, for $j \in \{1, \ldots, n-1\}$. Obviously, since $|\mathcal{I}| > 2^b$, there are more inputs than strategies to choose from and, thus, there

are two different inputs for one advice string. In order to be optimal, A needs to take the j-th and $(j+1)$-th item for the instance I_j and, hence, this choice is unique for every input from $\mathcal{I}$. For any other choice of items on the instance I_j, A achieves a gain of at most $\frac{1}{2}+\delta$, leading to a competitive ratio of

$$\frac{1}{\frac{1}{2}+\delta} = 2-\varepsilon$$

as we claimed. □

The competitive ratio that is achievable with respect to the number of used advice bits now makes a second jump as stated by the following theorem.

Theorem 6. *Let $\varepsilon > 0$. There exists an online algorithm* A *with advice for* SimpleKnapsack *that achieves a competitive ratio of $1+\varepsilon$ reading*

$$\left\lceil \frac{2\varepsilon+2}{\varepsilon} \right\rceil \cdot \lceil \log n \rceil + 2 \cdot \left\lceil \log \frac{2\varepsilon+2}{\varepsilon} \right\rceil + 2 \cdot \lceil \log \lceil \log n \rceil \rceil + 1$$

bits of advice.

Proof. Let $\varepsilon > 0$ be arbitrary, but fixed, and let $\delta = \varepsilon/(2+2\varepsilon)$. Suppose there does not exist any item within the input of size larger than δ which can be indicated using one bit at the beginning of the advice tape. Then, A may safely take sets greedily which leads to a competitive ratio of

$$\frac{1}{1-\delta} = 1 + \frac{\delta}{1-\delta} = 1 + \frac{\varepsilon}{2+\varepsilon} \leq 1+\varepsilon.$$

Now assume the contrary, i. e., there exist some items of size $> \delta$. The oracle inspects the optimal solution which consists of two disjoint sets of items S_1 and S_2, where S_1 denotes the set of i *heavy* items of size $> \delta$ and S_2 contains j *light* items of size $\leq \delta$. Let s_1 $[s_2]$ be the sum of all weights of the items in S_1 $[S_2]$. The indices of all heavy items are written onto the advice tape using $i \cdot \lceil \log n \rceil$ bits (also, we need to communicate i which can be done using another $\lceil \log 1/\delta \rceil$ bits). Since the sum of all weights of any solution does not exceed 1, we clearly have $i \leq 1/\delta$, i. e., i is constant with respect to n. For being able to decode the advice string, additionally the length $\lceil \log n \rceil$ of such an index has to be included in the advice in some self-delimiting form using $2\lceil \log \lceil \log n \rceil \rceil$ bits.[3]

Moreover, let the oracle encode a number k on the advice tape, where k is such that

$$k\delta \leq s_2 < (k+1)\delta.$$

Since A knows ε and therefore δ, it computes $k\delta$ and thus obtains a lower bound on s_2, i. e., the part of the solution that is due to the light items. Every such light item is taken as long as their sum is below $k\delta$. It is immediate that $k \leq 1/\delta$, due to $s_2 \leq 1$. According to Observation 1, A packs at least as many items from S_2 such that their sum is not smaller than $k\delta - \delta \geq s_2 - 2\delta$. Therefore, we get a competitive ratio of

$$\frac{s_1+s_2}{s_1+s_2-2\delta} \leq \frac{1}{1-2\delta} = 1 + \frac{2\delta}{1-2\delta} = 1+\varepsilon.$$

Since k is an integer from the range $0 \ldots 1/\delta$, it can be encoded using $\lceil \log(1/\delta) \rceil$ bits. The total number of advice bits used by the algorithm is

$$1 + i \cdot \lceil \log n \rceil + 2 \cdot \left\lceil \log \frac{1}{\delta} \right\rceil + 2 \cdot \lceil \log \lceil \log n \rceil \rceil \leq 1 + \frac{1}{\delta} \cdot \lceil \log n \rceil + 2 \cdot \left\lceil \log \frac{1}{\delta} \right\rceil + 2 \cdot \lceil \log \lceil \log n \rceil \rceil$$

$$\leq 1 + \left\lceil \frac{2\varepsilon+2}{\varepsilon} \right\rceil \cdot \lceil \log n \rceil + 2 \cdot \left\lceil \log \frac{2\varepsilon+2}{\varepsilon} \right\rceil + 2 \cdot \lceil \log \lceil \log n \rceil \rceil$$

as claimed, which concludes the proof. □

[3] For an example on how to construct such self-delimiting encodings, see, for example, the proof of Theorem 5 in [1].

3.2 Randomized Online Algorithms

In this section, we study the random bit complexity of the problem. At first, suppose we use the same algorithm as in Theorem 4, but guess the advice bit. Obviously, this algorithm, which we call B, is 2-competitive with probability 1/2 and not competitive with the same probability, that is, 4-competitive in expectation. This bound is tight as the next theorem shows.

Theorem 7. *The randomized online algorithm* B *for* SimpleKnapsack *cannot be better than* 4*-competitive in expectation.*

Proof. Let $\varepsilon < 1/6$. Consider three items of sizes

$$\frac{1}{2} - \varepsilon, 3\varepsilon, \frac{1}{2} - \varepsilon.$$

A greedy approach takes the first two items and therefore obtains a gain of $1/2 + 2\varepsilon$, whereas the algorithm that waits for an item of size $\geq 1/2$ gains nothing. Thus, B is c-competitive only for

$$c \geq \frac{1 - 2\varepsilon}{\frac{1}{2}\left(\frac{1}{2} + 2\varepsilon\right) + \frac{1}{2} \cdot 0} = 4 \cdot \frac{1 - 2\varepsilon}{1 + 4\varepsilon}.$$

Since ε can be arbitrarily small, no competitive ratio better than 4 can be reached. □

It seems somehow intuitively clear that randomization (the average over good and bad) is twice as bad as advice (*always* good). However, while this is right for this specific strategy, we get the following: Remarkably, randomization and advice are equally powerful for SimpleKnapsack when dealing with a small amount of either random or advice bits.

Theorem 8. *There exists a randomized online algorithm* R *for* SimpleKnapsack *that achieves an expected competitive ratio of* 2 *and that uses one random bit.*

Proof. Consider the following deterministic online algorithms A_1 and A_2; A_1 is the straightforward greedy algorithm for SimpleKnapsack. A_2 locally simulates A_1 and does not take any item until it realizes that an item just offered would not fit into A_1's solution anymore. A_2 then acts greedily starting from here. If the input consists of items that, in the sum, have a weight less than the knapsack's capacity, A_1 is obviously optimal, while A_2 might have gain zero. If, however, this is not the case, the gain of A_1 plus the gain of A_2 is at least 1.

Let R choose between A_1 and A_2 uniformly at random. Obviously, one random bit suffices to do that. We then immediately get that the expected gain of R is at least 1/2, and the competitive ratio of R is thus at most 2. □

Please note that the lower bound of 2 on the competitive ratio from algorithms with advice (see Theorem 5) carries over immediately for the randomized case. Therefore, Theorem 8 is tight. The above results imply that randomization and advice are equally powerful when we consider a sub-logarithmic number of bits.

As we have seen before (see Theorem 6), logarithmic advice helps a lot. On the other hand, we now show that this is not the case for randomization.

Theorem 9. *No randomized online algorithm for* SimpleKnapsack *can be better than* 2*-competitive (independent of the number of random bits the computation is based on).*

Proof. Consider the following class of inputs. At first, an item of weight $\varepsilon > 0$ is offered. After that, either nothing else is offered or an additional item of size 1.

Now consider any algorithm R that decides to use the first item with non-zero probability p (else, its gain is obviously zero). If R takes the item, of course, it cannot use the second one if it is offered. On the other hand, if R does not take the first item (with probability $1 - p$), it does not

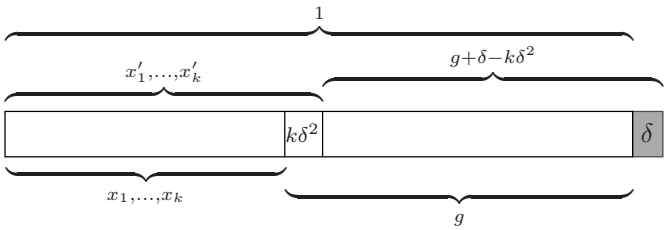

Fig. 1. The gains of both $\mathcal{O}pt$ and A

have any gain if there is no second item. Suppose the second item is offered. Algorithm R then has competitive ratio

$$\frac{1}{p \cdot \varepsilon + (1-p) \cdot 1},$$

and, if the second item is not offered, R has competitive ratio

$$\frac{\varepsilon}{p \cdot \varepsilon}.$$

By equalizing the ratios, we get

$$\frac{1}{(\varepsilon - 1) \cdot p + 1} = \frac{1}{p} \quad \Longleftrightarrow \quad p = \frac{1}{2-\varepsilon}$$

and, thus R is no better than $(2-\varepsilon)$-competitive. □

Let us summarize: With one random bit, we can achieve a (tight) bound of 2. However, any additional bit does not help at all.

3.3 Resource Augmentation

In this subsection, we allow the online algorithms considered to use more powerful resources than the optimal offline algorithm it is compared against. This model of *resource augmentation* was used for the online knapsack problem in [9] as well as for many other online problems, see, e. g., [4, 10, 15]. More precisely, we allow an online algorithm A to overpack the knapsack by some $\delta > 0$ whereas the optimal solution is merely allowed to fill it up to 1.

Theorem 10. *Let $1/4 > \delta > 0$. There exists an online algorithm* A *for* SimpleKnapsack *that achieves a competitive ratio of $1 + 3\delta/(1-4\delta)$ in the δ-resource-augmented model, using at most*

$$\left\lceil 2 \log \left\lceil \frac{1}{\delta} \right\rceil + \frac{1}{\delta} \cdot \log \left\lceil \frac{1}{\delta^2} \right\rceil \right\rceil + 1$$

advice bits.

Proof. Consider any instance I and let $\mathcal{O}pt = \texttt{Opt}(I)$ denote an optimal solution computed by an optimal offline algorithm Opt. Suppose $\text{cost}(\texttt{Opt}) \leq 1/2$. In this case, there is obviously no item with size $> 1/2$ and a simple greedy strategy enables A to be optimal. We fix the first advice bit to indicate whether $\mathcal{O}pt$ has size $1/2$ or smaller and, in the further analysis, assume the contrary.

To this end, let $\mathcal{O}pt = \{x_1, \ldots, x_k\} \dot\cup \{y_1, \ldots, y_m\}$ denote an optimal solution computed by an algorithm Opt where the items x_i have weights $\geq \delta$ and the items y_j have weights $< \delta$. Obviously, we have $k \leq 1/\delta$. A knows δ and is designed such that it reads all the approximate sizes (computed via an integer division by δ^2) of all *heavy* items and the fraction of the knapsack that is filled using *light* ones from the advice tape.

First, we show how the heavy items are encoded. To this end, let

$$\overline{x}_i := j \text{ such that } j \cdot \delta^2 \leq x_i < (j+1) \cdot \delta^2,$$

for every heavy item x_i. All $\overline{x}_i$s are sequentially written onto the advice tape and read by $\mathtt{A}$ right after the first offer. Thus, if $\mathtt{A}$ is offered any item x', it checks whether the corresponding $\overline{x}_i$ is part of the advice, that is, if there exists $\overline{x}_i$ such that $\delta^2 \cdot \overline{x}_i \leq x' < \delta^2 \cdot (\overline{x}_i + 1)$. If so, x' is taken into the knapsack as an element x'_i corresponding to x_i; else it is neglected. In the former case, there exists x_i that is part of $\mathcal{O}pt$ and $x_i - \delta^2 < x'_i < x_i + \delta^2$. Clearly, there are at most k different $\overline{x}_i$s (as many as there are corresponding heavy items) and each is at most of size $1/\delta^2$; hence, to communicate all of these values, we need no more than

$$k \cdot \log \left\lceil \frac{1}{\delta^2} \right\rceil \leq \left\lceil \frac{1}{\delta} \cdot \log \left\lceil \frac{1}{\delta^2} \right\rceil \right\rceil$$

advice bits, which is obviously constant with respect to n. However, to be able to decode the advice, $\mathtt{A}$ needs to know k beforehand.[4] The value of k can be written onto the advice tape in a self-delimiting form using another $2\lceil \log k \rceil \leq 2\lceil \log 1/\delta \rceil$ additional bits. The number of bits needed to encode the elements $\overline{x}_i$ can be calculated by $\mathtt{A}$ without any further knowledge.

Note that we have

$$-\delta + \sum_{i=1}^{k} x_i \leq \sum_{i=1}^{k} x'_i \leq k \cdot \delta^2 + \sum_{i=1}^{k} x_i \leq \delta + \sum_{i=1}^{k} x_i,$$

which means that, for every heavy item, $\mathtt{A}$ chooses an item such that the algorithm uses at most δ^2 more space within the knapsack. Thus, the sum of the chosen heavy items uses at most δ more space than the heavy items in $\mathcal{O}pt$.

We now distinguish two cases regarding the size of an optimal solution.

Case 1. Suppose that $\text{cost}(\mathcal{O}pt) < 1 - \delta$. This directly implies that all light items are part of the optimal solution, because, otherwise, $\mathcal{O}pt$ would not be optimal. Algorithm $\mathtt{A}$ uses at most δ more space for the heavy items as $\mathcal{O}pt$, hence it takes all light items as well. On the other hand, the sum of weights of the heavy items chosen by $\mathtt{A}$ is at least $-\delta + \sum_{i=1}^{k} x_i$, hence $\mathtt{A}$ has gain at least $\text{cost}(\mathcal{O}pt) - \delta$. Thus,

$$\text{comp}(\mathtt{A}(I)) \leq \frac{\text{cost}(\mathcal{O}pt)}{\text{cost}(\mathcal{O}pt) - \delta} \leq 1 + \frac{\delta}{\text{cost}(\mathcal{O}pt) - \delta} \leq 1 + \frac{2\delta}{1 - 2\delta},$$

where the last inequality follows from the fact that $\text{cost}(\mathcal{O}pt) \geq 1/2$.

Case 2. Suppose that $1 - \delta \leq \text{cost}(\mathcal{O}pt) \leq 1$. Let us now consider the light items. To this end, let $g := 1 - \sum_{i=1}^{k} x_i$ denote the space $\mathcal{O}pt$ is left with after packing all heavy items into the knapsack (see Fig. 1); $\mathtt{A}$ does not know g but calculates the approximate value

$$g' := 1 - \sum_{i=1}^{k} (\overline{x}_i + 1)\delta^2.$$

It follows that $g - \delta \leq g' \leq g$. Note that both $\mathtt{A}$ and $\mathcal{O}pt$ have all light items available. Now, for $z \in \{g, g'\}$, consider the instance $I(z)$ shrunk to a knapsack capacity of z and light items only and let $\mathcal{O}pt(z)$ denote the corresponding optimal solution for this instance. Since $\mathtt{A}$ acts greedily on $I(g')$, by Observation 1 it follows that $\mathtt{A}$ is either optimal or has a gain of at least $g' - \delta \geq g - 2\delta \geq \text{cost}(\mathcal{O}pt(g)) - 2\delta$. On the other hand, $\mathtt{OPT}$ obtains a gain of exactly $\text{cost}(\mathcal{O}pt(g))$ on $I(g)$. Thus,

$$\begin{aligned}
\text{comp}(\mathtt{A}(I)) = \frac{\text{cost}(\mathcal{O}pt)}{\text{cost}(\mathtt{A}(I))} &\leq \frac{\text{cost}(\mathcal{O}pt)}{\sum_{i=1}^{k} x'_i + g' - \delta} \leq \frac{\text{cost}(\mathcal{O}pt)}{\sum_{i=1}^{k} x_i + g - 2\delta} \\
&\leq \frac{\text{cost}(\mathcal{O}pt)}{\sum_{i=1}^{k} x_i + \text{cost}(\mathcal{O}pt(g)) - 3\delta} = \frac{\text{cost}(\mathcal{O}pt)}{\text{cost}(\mathcal{O}pt) - 3\delta} \leq 1 + \frac{3\delta}{1 - 4\delta},
\end{aligned}$$

which finishes the claim. □

[4] Recall that the length of the advice is not known to $\mathtt{A}$, but $\mathtt{A}$ reads the advice from an infinite tape.

4 The Weighted Case

We now consider the general knapsack problem, Knapsack, from Definition 3 where every item has both a weight and a value. However, our results only hold if we restrict ourselves to instances where the costs and weights can be represented within polynomial space. More formally, for any item x, let $w(x)$ be the weight of x, $c(x)$ be the cost of x and $r(x) := c(x)/w(x)$ be the ratio of its cost and weight. We assume that, for every x, $c(x)$ and $w(x)$ are rational numbers, and their numerators and denominators are bounded by $2^{p(n)}$ for some fixed polynomial $p(n)$.

First of all, we note that the lower bounds for SimpleKnapsack from the previous section carry over immediately, since we are now dealing with a generalization of the above problem. Second, Theorem 2 obviously also applies for the general knapsack problem.

Theorem 11. *No online algorithm for* Knapsack *using strictly less than* $\log n$ *bits of advice is competitive.*

Proof. Suppose that $\mathtt{A}$ reads $k < \log n$ advice bits which allows it to distinguish at most 2^k different inputs. We construct a set $\mathcal{I}$ of n different instances as follows. Let $\alpha := 2^n$ and let I_s be the instance determined by the items

$$(1, \alpha), (1, \alpha^2), \ldots, (1, \alpha^s), (1, 1), \ldots, (1, 1), (1, 1),$$

for $s \in \{1, \ldots, n\}$ and $\mathcal{I} = \{I_s \mid 1 \leq s \leq n\}$. Obviously, since $|\mathcal{I}| > 2^k$, there are more inputs than strategies to choose from and, thus, there are two different inputs for one advice string. Let these two instances be I_i and I_j and assume $i > j$. The unique optimal solution for I_i $[I_j]$ fills the knapsack with the i-th $[j$-th$]$ item yielding a gain of α^i $[\alpha^j]$.

Clearly, if $\mathtt{A}$ does not choose the j-th item when given the instance I_j, its gain is at least a factor of α away from $\mathcal{O}pt$. But this means that, since in the first j time steps, $\mathtt{A}$ cannot distinguish between I_i and I_j (and it is given the same fixed advice string), that $\mathtt{A}$ also takes the j-th item which results in a competitive ratio is at least $\alpha^i/\alpha^j \geq \alpha$ finishing the proof. □

In the following, we show how to solve the general knapsack problem almost optimally when using logarithmic advice. This implies that the bound from Theorem 11 is asymptotically tight.

Theorem 12. *Let* $\varepsilon > 0$. *There exists an online algorithm* $\mathtt{A}$ *with advice for* Knapsack *that achieves a competitive ratio of* $1 + \varepsilon$ *using at most* $\mathcal{O}(\log n)$ *bits of advice.*

Proof. Let $\delta = \sqrt{1+\varepsilon} - 1$. Consider any optimal solution $\mathcal{O}pt$ and let

$$c' := (1+\delta)^{\lfloor \log_{1+\delta}(\mathrm{cost}(\mathcal{O}pt)) \rfloor},$$

i. e., c' is an approximation of $\mathrm{cost}(\mathcal{O}pt)$ such that

$$\mathrm{cost}(\mathcal{O}pt)/(1+\delta) < c' \leq \mathrm{cost}(\mathcal{O}pt).$$

Next, let $x_1, \ldots, x_k$ be all items in $\mathcal{O}pt$ with cost at least $\delta \cdot c'$. Since there are at most $\mathrm{cost}(\mathcal{O}pt)/(\delta \cdot c')$ such items, we immediately get $k \leq (1+\delta)/\delta$.

Let $\mathcal{S}_1$ be an (offline) solution constructed as follows. At first, all "expensive" items $x_1, \ldots, x_k$ are taken; then, the rest of the knapsack is filled using items that have costs less than $\delta \cdot c'$ greedily by the ratio of their cost and weight in descending order.

Consider $\mathcal{S}_1$ plus the item x that is the first one that did not fit in the greedy phase of $\mathcal{S}_1$'s construction. $\mathcal{S}_1 \cup \{x\}$ has higher cost than $\mathcal{O}pt$. Since $c(x) \leq \delta \cdot c' \leq \delta \cdot \mathrm{cost}(\mathcal{O}pt)$, we get that

$$\mathrm{cost}(\mathcal{S}_1) \geq (1-\delta)\mathrm{cost}(\mathcal{O}pt).$$

Let $y_1, \ldots, y_l$ denote the items of $\mathcal{S}_1$ added in the greedy phase. Without loss of generality, assume that $r(y_1) \geq r(y_2) \geq \ldots \geq r(y_l)$ and let

$$r' := (1+\delta)^{\lceil \log_{1+\delta}(r(y_l)) \rceil},$$

i. e., r' is an approximation of $r(y_l)$ such that

$$r(y_l) \leq r' < r(y_l) \cdot (1+\delta).$$

Let m be the largest number such that $r(y_m) \geq r'$, that is, the items $r(y_1), \ldots, r(y_m)$ have ratios of at least r' and all other items have ratios between r' and $r'/(1+\delta)$. Let v be the space not occupied by $x_1, \ldots, x_k, y_1, \ldots, y_m$ in $\mathcal{S}_1$, i. e.,

$$v := 1 - \sum_{i=1}^{k} w(x_i) - \sum_{i=1}^{m} w(y_i).$$

Intuitively speaking, if we consider the part of the solution $\mathcal{S}_1$ that consists of the items y_i, for $i > m$, we see that this is a solution of an "almost-unweighted" knapsack instance with knapsack capacity v. Therefore, we can approximate it by a solution for the unweighted knapsack problem without much harm.

To this end, let

$$v' := (1+\delta)^{\lfloor \log_{1+\delta} v \rfloor},$$

i. e., v' is an approximation of v such that

$$v/(1+\delta) < v' \leq v.$$

Furthermore, let

$$\{z_1, \ldots, z_j\} = S := \{y_i \mid y_i \in \{y_{m+1}, \ldots, y_l\}, w(y_i) \geq \delta \cdot v'\},$$

that is, $z_1, \ldots, z_j$ are all items from $\mathcal{S}_1$ that have a ratio of roughly r' and whose weights are at least a δ-fraction of v'. Since $v' > v/(1+\delta)$, there are at most $(1+\delta)/\delta$ such items.

Again, we consider an (offline) solution $\mathcal{S}_2$, which is constructed as follows. At first, all items

$$x_1, \ldots, x_k, y_1, \ldots, y_m, z_1, \ldots, z_j$$

are taken. After that, we use all remaining items of weight less than $\delta \cdot v'$ and a ratio of at least $r'/(1+\delta)$; each of these items is greedily added to $\mathcal{S}_2$ if it fits. We now show that

$$\text{cost}(\mathcal{S}_2) \geq \frac{(1+\delta)^2}{1-\delta} \text{cost}(\mathcal{S}_1). \tag{1}$$

To this end, consider two cases. If the greedy construction of $\mathcal{S}_2$ takes all possible items, $\mathcal{S}_2$ contains all items included in $\mathcal{S}_1$, and Inequality (1) follows trivially. Therefore, we may assume the contrary.

Obviously, the cost of $\mathcal{S}_1$ is at most

$$\sum_{i=1}^{k} c(x_k) + \sum_{i=1}^{m} c(y_i) + \sum_{i=1}^{j} c(z_i) + v \cdot r' \leq \sum_{i=1}^{k} c(x_k) + \sum_{i=1}^{m} c(y_i) + \sum_{i=1}^{j} c(z_i) + v' \cdot (1+\delta) \cdot r'.$$

On the other hand, the cost of $\mathcal{S}_2$ is at least

$$\sum_{i=1}^{k} c(x_k) + \sum_{i=1}^{m} c(y_i) + \sum_{i=1}^{j} c(z_i) + v' \cdot (1-\delta) \cdot r'/(1+\delta),$$

because the greedy step packed items of total weight of at least $(1-\delta) \cdot v'$ with a ratio of at least $r'/(1+\delta)$. It follows that Inequality (1) holds.

Putting all together, we finally get

$$\text{cost}(\mathcal{S}_2) \geq \frac{(1+\delta)^2}{1-\delta} \text{cost}(\mathcal{S}_1) \geq (1+\delta)^2 \text{cost}(\mathcal{O}pt) = (1+\varepsilon)\mathcal{O}pt$$

as claimed.

Let us now look at the number of bits necessary to be communicated to `A`. At first, `O` needs to encode n and k which can be done using no more than $2\lceil\log\lceil\log n\rceil\rceil + 2\lceil\log n\rceil$ bits. Furthermore, since `A` knows δ, it suffices to read at most

$$\left\lceil \log \lfloor \log_{1+\delta} 2^{p(n)} \rfloor \right\rceil \leq \log\left(\frac{\log 2^{p(n)}}{\log(1+\delta)}\right) + 1 \in \mathcal{O}\left(\log n^d\right)$$

advice bits to communicate c', where d is the degree of the polynomial $p(n)$. We immediately see that, to encode r' and v', we also need no more than $\mathcal{O}\left(\log n^d\right)$ bits. The indices of the items x_i can be specified using $k\lceil\log n\rceil \leq (1+\delta)/\delta \log n + 1$ additional bits. Similarly, the indices of the items z_i can be communicated using $j\lceil\log n\rceil \leq (1+\delta)/\delta \log n + 1$ bits.

We conclude that at most $\mathcal{O}\left(\log n^d\right) = \mathcal{O}(\log n)$ bits are needed in total. Finally, the online algorithm `A` works as follows to construct $\mathcal{S}_2$ using the advice as specified above.

Algorithm `A`

1. **for** any x **do**
2. **if** $x = x_i$ for some i, **use;**
3. **else if** $c(x) \geq \delta \cdot c'$, **discard;**
4. **else if** $r(x) \geq r'$, **use;**
5. **else if** $x = z_i$ for some i, **use;**
6. **else if** $r(t) < r'/(1+\delta)$ **or** $w(t) \geq \delta \cdot v'$, **discard;**
7. **else if** total weight of all items taken at line 7 $\leq v'$, **use;**
8. **else discard;**
9. **end**

This finishes our proof. □

5 Conclusion

We have analyzed the advice complexity and the random bit complexity of the online knapsack problem. For the unweighted case, the advice complexity exhibits a very interesting phase transition: Less than $\log(n-1)$ advice bits do not improve over a single bit of advice, but $\mathcal{O}(\log n)$ advice bits already allow for an almost optimal competitive ratio. A similar phenomenon can be observed for the random bit complexity. Here, a single random bit achieves a competitive ratio of 2 and no additional randomness can improve this result. We have also seen that, when allowing online algorithms to overpack the knapsack a little bit, a constant number of advice bits suffices to produce an output that is arbitrarily close to the optimum. Finally, we have shown that $\mathcal{O}(\log n)$ bits are also sufficient to get arbitrarily close to an optimal solution for the weighted online knapsack problem. Here, the $\mathcal{O}$-notation hides a larger constant as for the unweighted case.

For further research, it would be interesting to see how randomized online algorithms with advice behave on this problem, i. e., whether some of the $\mathcal{O}(\log n)$ advice bits in the proof of Theorem 6 can be substituted by some amount of random bits.

References

1. Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Královič, and Richard Královič. On the advice complexity of the k-server problem. In *Proc. of the 38th International Colloquium on Automata, Languages and Programming (ICALP 2011)*, volume 6755 of *Lecture Notes in Computer Science*, pages 207–218. Springer-Verlag, 2011.
2. Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Královič, Richard Královič, and Tobias Mömke. On the advice complexity of online problems. In *Proc. of the 20th International Symposium on Algorithms and Computation (ISAAC 2009)*, volume 5878 of *Lecture Notes in Computer Science*, pages 331–340. Springer-Verlag, 2009.

3. Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
4. János Csirik and Gerhard J. Woeginger. Resource augmentation for online bounded space bin packing. *Journal of Algorithms*, 44(2):308–320, 2002.
5. Stefan Dobrev, Rastislav Královič, and Dana Pardubská. Measuring the problem-relevant information in input. *Theoretical Informatics and Applications (RAIRO)*, 43(3):585–613, 2009.
6. Yuval Emek, Pierre Fraigniaud, Amos Korman, and Adi Rosén. Online computation with advice. *Theoretical Computer Science*, 412(24):2642–2656, 2011.
7. Juraj Hromkovič. *Algorithmics for Hard Problems*. Springer-Verlag, second edition, 2004.
8. Juraj Hromkovič, Rastislav Královič, and Richard Královič. Information complexity of online problems. In *Proc. of the 35th International Symposium on Mathematical Foundations of Computer Science (MFCS 2010)*, volume 6281 of *Lecture Notes in Computer Science*, pages 24–36. Springer-Verlag, 2010.
9. Kazuo Iwama and Guochuan Zhang. Online knapsack with resource augmentation. *Information Processing Letters*, 110(22):1016–1020, 2010.
10. Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *Journal of the ACM*, 47(4):617–643, 2000.
11. Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems*. Springer-Verlag, 2004.
12. Dennis Komm and Richard Královič. Advice complexity and barely random algorithms. *Theoretical Informatics and Applications (RAIRO)*, 45(2):249–267, 2011.
13. Dennis Komm, Richard Královič, and Tobias Mömke. On the advice complexity of the set cover problem. Technical Report 738, ETH Zurich, 2011.
14. Alberto Marchetti-Spaccamela and Carlo Vercellis. Stochastic on-line knapsack problems. *Mathematical Programming*, 68:73–104, 1995.
15. Cynthia A. Phillips, Clifford Stein, Eric Torng, and Joel Wein. Optimal time-critical scheduling via resource augmentation. *Algorithmica*, 32(2):163–200, 2002.
16. Yunhong Zhou, Deeparnab Chakrabarty, and Rajan M. Lukose. Budget constrained bidding in keyword auctions and online knapsack problems. *Proc. of the 4th International Workshop on Internet and Network Economics (WINE 2008)*, volume 5385 of *Lecture Notes in Computer Science*, pages 566–576. Springer-Verlag, 2008.