Online Simple Knapsack with Reservation Costs

Hans-Joachim Böckenhauer
Department of Computer Science, ETH Zürich, Switzerland

Elisabet Burjons
Department of Computer Science, RWTH Aachen, Germany

Juraj Hromkovič
Department of Computer Science, ETH Zürich, Switzerland

Henri Lotze
Department of Computer Science, RWTH Aachen, Germany

Peter Rossmanith
Department of Computer Science, RWTH Aachen, Germany

Abstract

In the Online Simple Knapsack Problem we are given a knapsack of unit size 1. Items of size smaller or equal to 1 are presented in an iterative fashion and an algorithm has to decide whether to permanently reject or include each item into the knapsack without any knowledge about the rest of the instance. The goal is then to pack the knapsack as full as possible. In this work, we introduce a third option additional to those of packing and rejecting an item, namely that of reserving an item for the cost of a fixed fraction α of its size. An algorithm may pay this fraction in order to postpone its decision on whether to include or reject the item until after the last item of the instance was presented.

While the classical Online Simple Knapsack Problem does not admit any constantly bounded competitive ratio in the deterministic setting, we find that adding the possibility of reservation makes the problem constantly competitive, with varying competitive ratios depending on the value of α. We give upper and lower bounds for the whole range of reservation costs, with tight bounds for costs up to $1/6$ – an area that is strictly 2-competitive – , for costs between $\sqrt{2}-1$ and 1 – an area that is strictly $(2+\alpha)$-competitive up to $\phi-1$, and strictly $1/(1-\alpha)$-competitive above $\phi-1$, where ϕ is the golden ratio.

With our analysis, we find a counterintuitive characteristic of the problem: Intuitively, one would expect that the possibility of rejecting items becomes more and more helpful for an online algorithm with growing reservation costs. However, for higher reservation costs above $\sqrt{2}-1$, an algorithm that is unable to reject any items tightly matches the lower bound and is thus the best possible. On the other hand, for any positive reservation cost smaller than $1/6$, any algorithm that is unable to reject any items performs considerably worse than one that is able to reject.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Online problem, Simple knapsack, Reservation costs

Digital Object Identifier 10.4230/LIPIcs.STACS.2021.16

Related Version *Full Version*: `https://arxiv.org/abs/2009.14043`

Acknowledgements We want to thank the anonymous reviewers for pointing out imprecise formulations and helping to improve the overall quality of this work.

1 Introduction

Online algorithms can be characterized by receiving their input in an iterative fashion and having to act in an irrevocable way on each piece of the input, e. g., by deciding whether to include an element into the solution set or not. This has to be done with no additional knowledge about the contents or even the length of the rest of the instance that is still

to be revealed. The goal, as with regular offline algorithms, is to optimize some objective function. In order to measure the performance of an online algorithm, it is compared to that of an optimal offline algorithm on the same instance. The worst-case ratio between the performances of these algorithms over all instances is then called the *strict competitive ratio* of an online algorithm, which was introduced by Sleator and Tarjan [22].

One of the arguably most basic and famous problems of online computation is called the *ski rental problem* [19]. In this problem, someone is going on a skiing holiday of not yet fixed duration without owning the necessary equipment. On each day, she decides, based solely on that day's short-term weather report, whether skiing is possible on that day or not. On each day with suitable weather, she can either buy the equipment or rent it for that day for a fixed percentage of the cost of buying a pair of ski. Arguably, the only interesting instances are those in which a selected number of days are suitable for skiing, followed by the rest of the days at which she is unable to go skiing anymore. This is simply due to the fact that, as long as she has not decided to buy a pair of ski yet, a day at which no skiing is possible requires no buy-or-rent decision. Thus, the problem can be simplified as follows: The input is a string of some markers that represent days suitable for skiing, and as soon as the instance ends, skiing is no longer possible.

This notion of delaying a decision for a fixed percentage of the cost is the model that we want to study in this work. Note that, while the ski-rental problem in its above-mentioned form only models a single buy-or-rent decision (buying or renting a single commodity), iterated versions of it have been discussed in the literature, with important applications, e. g., to energy-efficient computing [15, 1, 4]. In the following, we investigate the power of delaying decisions for another more involved problem, namely the online knapsack problem.

The *knapsack problem* is defined as follows. Given a set of items I, a size function $w\colon I \to \mathbf{R}$ and a gain function $g\colon I \to \mathbf{R}$, find a subset $S \subseteq I$ such that the sum of sizes of S is lower or equal to the so-called *knapsack capacity* (which we assume to be normalized to 1 in this paper) and the sum of the gains is maximized. The online variant of this problem reveals the items of I piece by piece, with an algorithm having to immediately decide whether to pack a revealed item or to discard it.

The knapsack problem is a classical hard optimization problem, with the decision variant being shown to be NP-complete as one of Karp's 21 NP-complete problems [18]. The offline variant of this problem was studied extensively in the past, showing for example that it admits a fully polynomial time approximation scheme [14].

A variant of this problem in which the gains of all items coincide with their respective sizes is called the *simple knapsack problem*. Both variants do not admit a constantly bounded competitive ratio [20]. In this paper, we focus on the online version of the latter variant, which we simply refer to as the *online knapsack problem*, short OnlineKP, if not explicitly stated otherwise.

We propose a rather natural generalization of the online knapsack problem. Classically, whenever an item of the instance is presented, an online algorithm has to irrevocably decide whether to include it into its solution (i. e., pack it into the knapsack) or to discard it. In our model, the algorithm is given a third option, which is to reserve an item for the cost of a fixed percentage $0 \leq \alpha \leq 1$ of its value. The algorithm may reserve an arbitrary number of items of the instance and decide to pack or discard them at any later point. Philosophically speaking, the algorithm may pay a prize in order to (partially) "go offline." It is easy to see that, for $\alpha = 0$, the complete instance can be learned before making a decision without any disadvantages, essentially making the problem an offline problem, while, for $\alpha = 1$, reserving an item is not better than discarding it because packing a reserved item does not add anything to the gain.

One of the key properties of our reservation model applied to the simple knapsack problem is its unintuitive behavior with respect to rejecting items. It shows some sharp thresholds for the competitive ratio at seemingly arbitrary points: For some interval of low reservation fees, the competitive ratio is not affected by the charge at all, on the next interval, it depends linearly on the charge, and on the last interval, the competitive ratio grows even faster. This behavior is further discussed in the following subsection.

Moreover, this extension to the knapsack problem is arguably quite natural: Consider somebody trying to book a flight with several intermediate stops. Since flight prices are subject to quick price changes, it might be necessary to invest some reservation costs even for some flights not ending up in the final journey. The knapsack problem with reservations might also be seen as a simple model for investing in a volatile stock market, where a specific type of deviates is available (at some cost) that allows the investor to fix the price of some stock for a limited period of time.

The OnlineKP has been extensively studied under many other variations, including buffers of constant size in which items may be stored before deciding whether to pack them, studied by Han et al. [12]. Here, the authors allow for a buffer of size at least the knapsack capacity into which the items that are presented may or may not be packed and from which a subset is then packed into the actual knapsack in the last step. They extensively study the case in which items may be irrevocably removed from the buffer. Our model can be understood as having an optional buffer of infinite size, with the caveat that each buffered item induces a cost. A variant without an additional buffer, but with the option to remove items from the knapsack at a later point was studied by Iwama et al. [16]. The same model with costs for each removal that are proportional to a fraction f of each item from the knapsack was researched by Han et al. [11], which is closer to our model. Their model allows an algorithm to remove items that were already packed into the knapsack, for a proportion of the value of these items. However, we do not know of any simple reduction from one model to the other, which is supported by the considerably different behavior of the competitive ratio relative to the reservation cost. For OnlineKP, Han et al. show that the problem is 2-competitive for a removal cost factor $f \leq 0.5$ and becomes $(1 + f + \sqrt{f^2 + 2f + 5})/2$-competitive for $f > 0.5$.

Other models of the (simple) online knapsack problem have been studied as well, such as the behavior of the resource-augmentation model, where the knapsack of the online algorithm is slightly larger than the one of the offline algorithm on the same instances, studied by Iwama et al. [17]. When randomization is allowed, the OnlineKP becomes 2-competitive using only a single random bit and does not increase its competitive ratio with any additional bits of randomization [6]. A relatively young measure of the informational complexity of a problem is that of advice complexity, introduced by Dobrev, Královič, and Pardubská [9], revised by Hromkovič et al. [13] and refined by Böckenhauer et al. [5]. In the standard model, an online algorithm is given an advice tape of bits that an oracle may use to convey some bits of information to an online algorithm about the instance in order to improve its performance. Not surprisingly, a single advice bit also improves the competitive ratio to 2, but interestingly, any advice in the size of $o(\log n)$ advice bits does not improve this ratio [6].

1.1 Our Contributions

We study the behavior of the knapsack problem with reservation costs for a reservation factor $0 < \alpha < 1$. We analyze several subintervals for α separately with borders at $1/6$, $\alpha_0 = 1 - \frac{\sqrt{2}}{2} \approx 0.293$, $\alpha_1 \approx 0.318$, $\alpha_2 \approx 0.365$, $\alpha_3 = \sqrt{2} - 1 \approx 0.414$, and $\phi - 1 \approx 0.618$,

Table 1 Competitive ratios proven in this work.

α	Lower bound		Upper bound	
$0 < \alpha \leq \frac{1}{6}$	2	Thm 3	2	Thm 1
$\frac{1}{6} < \alpha \leq 1 - \frac{\sqrt{2}}{2}$	2	Thm 3	$2+\alpha$	Thm 5
$1 - \frac{\sqrt{2}}{2} < \alpha \leq \approx 0.318$	$\frac{1}{(1-\alpha)^2}$	Thm 4	$2+\alpha$	Thm 5
$\approx 0.318 < \alpha \leq \approx 0.365$	≈ 2.15	Thm 4	$2+\alpha$	Thm 5
$\approx 0.365 < \alpha \leq \sqrt{2}-1$	$\frac{1+\alpha}{1-\alpha}$	Thm 4	$2+\alpha$	Thm 5
$\sqrt{2}-1 < \alpha < \phi - 1$	$2+\alpha$	Thm 12	$2+\alpha$	Thm 5, 6
$\phi - 1 \leq \alpha < 1$	$\frac{1}{1-\alpha}$	Thm 15	$\frac{1}{1-\alpha}$	Thm 14

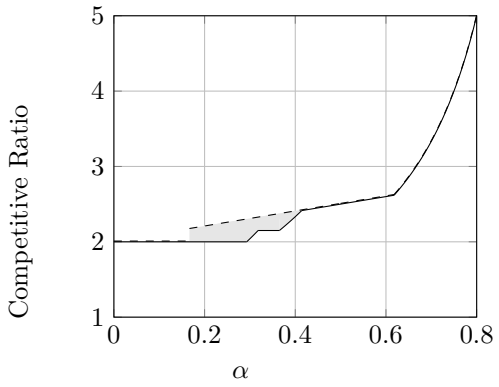

Figure 1 A schematic plot of the results proven in this paper. The full lines represent lower bounds while the dashed lines represent upper bounds.

where ϕ is the golden ratio. The bounds that we are providing, which are also illustrated in Figure 1, can be found in Table 1. We prove tight competitive ratios for $\alpha \leq \frac{1}{6}$ as well as for $\alpha \geq \sqrt{2} - 1$.

We also take a look at a subclass of algorithms for this problem that never discard presented items up to a point where they stop processing the rest of the input, with arguably paradox results: One would expect that, for very small reservation costs, reserving items instead of rejecting them right away is more helpful than for large reservation costs and that rejecting becomes more and more helpful with increasing reservation costs. But we prove that, while, for small values of α, every algorithm that is unable to reject an item is strictly dominated by algorithms that are able to reject items, for larger values of α this does not appear to be the case, with our algorithms used for $\alpha \geq \sqrt{2} - 1$ being nonrejecting and matching their lower bounds.

We cannot give a definitive answer on why this behavior can be observed. Our intuition is that for higher reservation costs, the behavior is more similar to the model without reservations. In this classical model, any errors that occur by ultimately not packing an item are punished severely, so while reserving everything is costly, it is not worse than rejecting any items.

The remainder of this paper is structured as follows: In Section 2, we present tight upper and lower bounds for small values of α as well as almost tight lower bounds for $\frac{1}{6} < \alpha < \sqrt{2} - 1$. In Section 3, we look at values of α up to $\phi - 1$, providing tight bounds for the complete interval. We conclude the competitive analysis with lower and upper bounds

for larger values of α in Section 4. Section 5 is devoted to a discussion of algorithms that are unable to reject items. We conclude the paper in Section 6. A full version that contains all the proofs is available via arXiv [3].

1.2 Preliminaries

Our model can be defined as follows. Consider a knapsack of size 1 and a *reservation factor* α, with $0 \leq \alpha \leq 1$. A *request sequence* I is a list of item sizes $x_1, x_2, \ldots, x_n$ with $x_i \leq 1$ for $1 \leq i \leq n$, which arrive sequentially. At time step i, the knapsack is filled up to some size $t_i \leq 1$ and the reserved items add up to size r_i. When an item with size x_i arrives, an online algorithm may *pack* this item into the knapsack if $x_i + t_i \leq 1$, it may also *reject* the item, or *reserve* the item at cost $\alpha \cdot x_i$.

At step $n+1$, no new items arrive and the knapsack contains all items that were taken with total size t_n and the reserved items have size $r_n = R$. An algorithm can additionally pack any of the reserved items which still fit into the knapsack, up to some size $t \leq 1$. The *gain* of an algorithm $\mathtt{A}$ solving ReserveKP with a reservation factor α on an instance I is $\text{gain}_{\mathtt{A}}(I) = t - \alpha \cdot R$.

The *strict competitive ratio* of an algorithm $\mathtt{A}$ on an instance I is, given a solution with optimal gain $\text{gain}_{\mathtt{OPT}}(I)$ on this instance, $\rho_{\mathtt{A}}(I) = \text{gain}_{\mathtt{OPT}}(I)/\text{gain}_{\mathtt{A}}(I)$. The general strict competitive ratio of an algorithm $\mathtt{A}$ is taken as the worst case over all possible instances, $\rho_{\mathtt{A}} = \max_I\{\rho_{\mathtt{A}}(I)\}$.

The strict competitive ratio as defined above is a special case of the well-known *competitive ratio* which relaxes the above definition by allowing for a constant additive term in the definition. Note that this generalized definition only makes sense for online problems in which the optimal solution has unbounded gain. Since the gain of an optimal solution for ReserveKP is bounded by the knapsack capacity, we only work with strict competitive ratios in this paper and will simply call it *competitive ratio* from now on. For a thorough introduction to competitive analysis of online algorithms, see the textbooks by Borodin and El-Yaniv [7] and by Komm [19].

2 Small Reservation Costs

In this section, we analyze the case of reservation costs below $\frac{1}{6}$. Additionally, we provide lower bounds for values of α up to $\sqrt{2}-1$. The upper bounds left out in this section are a byproduct of the upper bounds of the next section for medium reservation costs and can be found there.

2.1 Upper Bound for $0 < \alpha \leq \frac{1}{6}$

▶ **Theorem 1.** *ReserveKP has a competitive ratio of at most* 2 *if* $0 < \alpha \leq \frac{1}{6}$.

To prove Theorem 1, we need to start with some technical considerations. First, we define a threshold μ that will be used prominently in our online algorithm in the proof of Theorem 1. We choose μ exactly so large, that packing items of size μ yields a competitive ratio of at most 2, even if we have to pay reservation costs for all those items. We define $\mu = 1/(2(1-\alpha))$.

▶ **Lemma 2.** *Let* $\alpha \leq \frac{1}{2}$. *If an algorithm has reserved items of total size* $R \leq \mu$, *then filling the knapsack up to* μ *is sufficient to achieve a competitive ratio of at most* 2.

Algorithm 1 The case $0 < \alpha \leq \frac{1}{6}$.

```
1:  R := 0;
2:  for k = 1, ..., n do
3:      if x_k ≥ μ then pack x_k and stop
4:      else if μ ≤ x_k + R ≤ 1 then pack x_k and all reserved items and stop
5:      else if R ≤ 1 − μ then
6:          if x_k ≥ 1/2 then pack x_k and all reserved items and stop
7:          else reserve x_k and let R := R + x_k
8:      else if x_k + R < μ then reserve x_k and let R := R + x_k
9:      else
10:         if x_k ≥ 1/2 then
11:             if gain_OPT(x_k, R) ≥ μ then pack optimally and stop
12:             else reject x_k
13:         else pack x_k and reserved objects optimally and stop
14: pack the reserved items optimally
```

For small reservation costs, we design Algorithm 1, which unfortunately has a lot of case distinctions, most of which serve "easy" cases. The analysis of Algorithm 1 will prove Theorem 1. Let us look at some easy examples how the algorithm proceeds. If we feed a stream of small items of identical size $\varepsilon \ll \alpha$ into the algorithm, it will reserve all of them until their total size reaches μ, when all items are put into the knapsack and the algorithm stops in line 4. The gain of the algorithm is then at least $\mu - \alpha\mu$, which yields a competitive ratio of at least $\frac{1}{2}$.

In the next example, the reservation is still small, smaller than $1-\mu$, and a request comes for an item that is larger than $\frac{1}{2}$. If the item is larger than μ, it will be packed in line 3, and achieve the desired competitive ratio due to Lemma 2. Otherwise, if it is smaller than μ, but it reaches μ together with the reservation, it will be packed in line 4, and will still achieve the desired competitive ratio, due to the same lemma, otherwise, it will be packed in line 6. Because the reservation is smaller than $1 - \mu$, the item fits in with the reserved items, the only concern is that it does not reach up to μ and we cannot apply the lemma, however, the desired competitive ratio is still achieved as the item is packed without being reserved first, making the total $\text{gain}_{\mathtt{A}}(I)$ larger than $\frac{1}{2}$ as a result.

As a last example, we consider what happens when a large item (larger than $\frac{1}{2}$) arrives after the reservation size is already larger than $1 - \mu$, in this case, the only possibility that the item gets rejected is if the condition in line 11 is not satisfied. But literally this condition only requires that the optimal packing of the reserved items and the new item achieves the desired competitive ratio. Thus, we will have to make sure that the algorithm does not perform too badly by rejecting objects if the end of the request sequence is achieved and the algorithm is forced to pack in line 14.

Proof of Theorem 1. There are six ways how the algorithm might terminate. We have to prove for all six cases that the competitive ratio is at most 2, when the algorithm terminates at this point. These six possibilities correspond to lines 3, 4, 6, 11, 13, and 14.

Given a request x_k larger than μ, we know by Lemma 2 that we achieve the desired competitive ratio by packing x_k alone, which is done in line 3. If our request is smaller than μ, but $\mu \leq x_k + R \leq 1$, we again achieve the desired competitive ratio by packing all of the reserved objects together with x_k, thus satisfying the conditions for Lemma 2, which is carried out in line 4. Observe, for these two cases, that we do not reserve anything that would make R larger than μ.

Now, in line 5, if $R \leq 1-\mu$ and $x_k \geq \frac{1}{2}$, we know, since line 3 did not trigger the packing, that $x_k < \mu$ and all reserved items fit in the knapsack together with x_k. We only need to ensure that they achieve the desired competitive ratio, when the algorithm stops in line 6:

$$\frac{1}{x_k + (1-\alpha)R} \leq \frac{1}{1/2 + (1-\alpha)R} \leq 2 \,.$$

If line 9 of the algorithm is reached, i. e., $x_k + R > 1$ and $R > 1-\mu$, we can have the case that $x_k \geq \frac{1}{2}$. In this case, line 11 will ensure that the algorithm only stops and packs when the desired competitive ratio is achieved.

On the other hand, if $x_k + R > 1$, $R > 1-\mu$, and $x_k < \frac{1}{2}$, i. e., if the algorithm has reached line 13, we know, by inspecting the algorithm, that every object in R is smaller than $\frac{1}{2}$, so each of them fits into the knapsack together with x_k. Indeed, otherwise one of the items $x_1, \ldots, x_{k-1}$ would have triggered the packing in line 4 or 6. If we were not able to fill the knapsack up to μ with x_k and the reserved objects, it would mean that there is a reserved object $x_i > 1-\mu$ that does not fit in the end. However, this object fits into the knapsack together with x_k. We distinguish two cases. If $x_k \geq 1-\mu$, we put these two objects into the knapsack and achieve a gain of $x_k + x_i \geq 1-\mu+1-\mu = 2-2\mu$, and we know that $2-2\mu \geq \mu$ for every $\alpha \leq \frac{1}{4}$, so we achieve a competitive ratio of 2 by Lemma 2 when the algorithm terminates in line 13. Otherwise, if $x_k < 1-\mu$, and packing x_i and x_k into the knapsack, together with some other available reserved items still does not fill the knapsack up to μ, there must be yet another object x_j in the reservation $x_j > 1-\mu$, and by the same analysis $x_i + x_j > \mu$ for $\alpha \leq \frac{1}{4}$, achieving yet again the desired competitive ratio.

Now we are left with only one case to analyze: The algorithm reaches the end of the request sequence I without having stopped. Then it packs an optimal subset of the reserved items and ends in line 14. If there is an optimal solution that does not contain any items that were rejected by the algorithm, then our algorithm can pack the same items as in the optimal solution and the gain is $(1-\alpha)\text{gain}_{\texttt{OPT}}(I)$, which leads to a competitive ratio of

$$\frac{\text{gain}_{\texttt{OPT}}(I)}{(1-\alpha)\text{gain}_{\texttt{OPT}}(I)} = \frac{1}{1-\alpha} \leq 2 \quad \text{for } \alpha \leq \tfrac{1}{2}.$$

The optimal solution might include one of the rejected items, but only one, as all rejected items are larger than $\frac{1}{2}$. In this case, we know that the optimal solution contains the rejected item x_{rej} and some other items smaller than $\frac{1}{2}$ that will necessarily be reserved, that is, the size of an optimal solution is $x_{rej} + R'$ where $R' \subseteq R$. However, we also know that the rejected item did not fit with all reserved items when it appeared. This means that there is one reserved object x_i, of size $1-\mu < x_i < \frac{1}{2}$, and this object is not in R', as it does not fit with x_{rej}: otherwise it would have been taken by the algorithm, which then would stop, as two objects greater than $1-\mu$ that fit into the knapsack are enough to achieve a competitive ratio of 2 for $\alpha \leq \frac{1}{4}$ as we just saw in the previous case. Hence, the algorithm can pack at least $x_i + R'$, as $x_i < \frac{1}{2}$ and $x_i \notin R'$. Thus, the competitive ratio in this case is

$$\rho_{\texttt{A}} \leq \frac{x_{rej} + R'}{x_i + R' - \alpha R} \leq \frac{\mu + R'}{1 - \mu + R' - \alpha\mu} \,,$$

where the last inequality follows since $x_{rej} < \mu$ (otherwise, the algorithm would have terminated in line 3) and $R < \mu$ (otherwise, the algorithm would have stopped in line 4). We know that $0 \leq R' \leq 1-\mu$, but for the larger values of R' we know that, if at some point $x_i + R' \geq \mu$, the algorithm would have terminated sooner and not reached the end of the sequence. Therefore $R' \leq \mu - x_i \leq \mu - (1-\mu) \leq 2\mu - 1$ and $0 \leq R' \leq 2\mu - 1$. Then,

$$\frac{\mu + R'}{1 - \mu + R' - \alpha\mu} \leq \frac{\mu}{1 - \mu - \alpha\mu} \leq 2 \,,$$

for $0 \leq \alpha \leq \frac{1}{6}$, as can be shown with standard methods from calculus. ◀

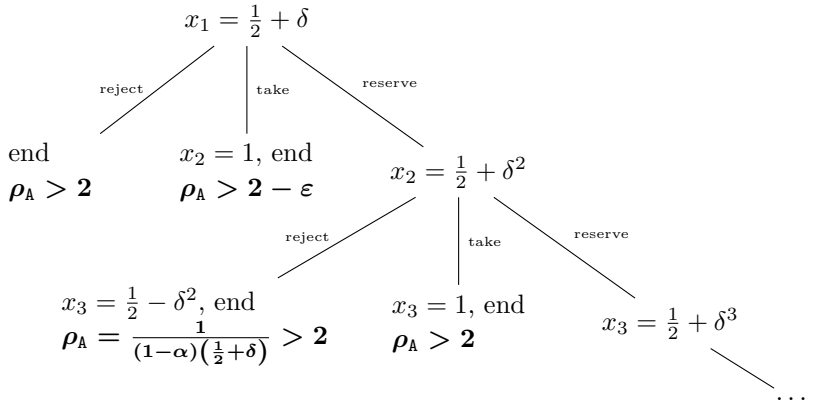

Figure 2 Sketch of the adversarial strategy that is used in the proof of Theorem 3.

2.2 Lower Bound for $0 < \alpha \leq \sqrt{2} - 1$

First we present an adversarial strategy that works for all values of α. Then we proceed to analyze the case where only three objects are presented as a generic adversarial strategy and find improved lower bounds for some values of α.

▶ **Theorem 3.** *For $\alpha > 0$ there exists no algorithm for reservation knapsack achieving a competitive ratio better than 2.*

Proof. Consider the following set of adversarial instances depicted in Figure 2. Given any $\varepsilon > 0$, the adversary presents first an object of size $\frac{1}{2} + \delta$ with $0 < \delta \ll \varepsilon$. If an algorithm takes this object, an object of size 1 will follow, making its competitive ratio $1/(\frac{1}{2}+\delta) > 2-\varepsilon$. If an algorithm rejects this object, no more objects will follow and it will not be competitive. If an algorithm reserves this object, then an object of size $\frac{1}{2} + \delta^2$ will be presented. Observe, that these two objects do not fit together into the knapsack. If an algorithm takes this object, an object of size 1 will be presented, and again the algorithm will achieve a competitive ratio worse than $2 - \varepsilon$. If an algorithm rejects this object, then an object of size $\frac{1}{2} - \delta^2$ will be presented. This object does not fit in the knapsack with the first one, thus the algorithm can only pack the first object, obtaining a competitive ratio worse than $2 - \varepsilon$. If an algorithm reserves it instead, an object of size $\frac{1}{2} + \delta^3$ will be presented. The adversary can follow this procedure on and on, and in each step the competitive ratios for algorithms that accept or reject the offered item only get worse due to the additional reservation costs.

The adversary can stop offering items as soon as the reservation costs are such that filling the knapsack will only result in a competitive ratio worse than 2. This shows that, for every $\varepsilon > 0$, the competitive ratio is at least $2 - \varepsilon$, so the best competitive ratio is at least 2. ◀

The adversarial strategy depicted in Figure 2 works for every positive value of α. Observe that if an algorithm continues to reserve items, the cumulated reservation cost will eventually exceed any remaining gain.

This strategy provides us with a lower bound that does not match the upper bound for $\alpha > \frac{1}{6}$. We improve the lower bound for larger α by designing a generic adversarial strategy with three items shown in Figure 3, which shows the competitive ratios for all possible outcomes. It is therefore bounded by

$$\rho_A \geq \min\left\{\frac{1}{s},\ \frac{1}{t-\alpha s},\ \frac{t}{(1-\alpha)t-\alpha s},\ \frac{t}{s-\alpha s}\right\}. \tag{1}$$

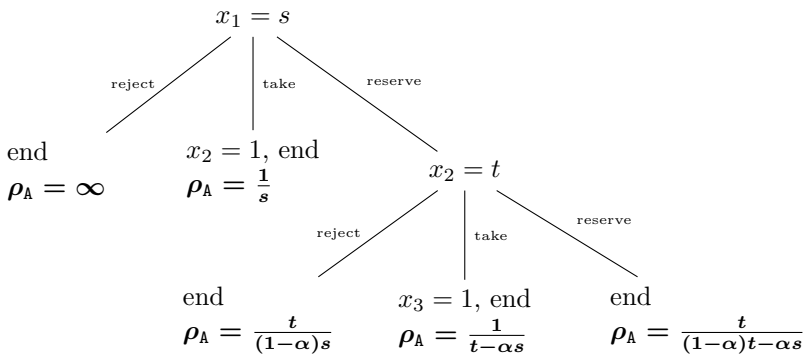

Figure 3 Diagram of a generic adversarial strategy with 3 items.

To prove a lower bound for every $0 < \alpha < 1$, we can choose s and t in order to make (1) as large as possible. Standard calculus leads to the bounds in the following theorem.

▶ **Theorem 4.** *The competitive ratio of* ReserveKP *is at least*
(a) $1/(1-\alpha)^2 \approx 2 \ldots 2.15$, *for* $0.293 \leq \alpha < 0.318$;
(b) $(2-\alpha_1)/(1-\alpha_1+\alpha_1^2) \approx 2.15$, *for* $0.318 \leq \alpha < 0.365$ *(recall that* $\alpha_1 \approx 0.318$*)*;
(c) $(1+\alpha)/(1-\alpha) \approx 2.15 \ldots 2.41$, *for* $0.365 \leq \alpha < 0.414$; *and*
(d) $2+\alpha \approx 2.41 \ldots 3$, *for* $0.414 \leq \alpha \leq 1$.

3 Medium Reservation Costs

We now consider the case of medium reservation costs, i. e., $\sqrt{2}-1 \leq \alpha < \phi - 1 = \frac{\sqrt{5}}{2} - \frac{1}{2}$. We will provide an upper bound of $2+\alpha$ for $\alpha < \phi - 1$ in Theorems 5 and 6. We complement these upper bounds by a tight lower bound of $2+\alpha$ for the whole range of medium-size reservation cost.

3.1 Upper Bound for $\frac{1}{6} < \alpha < \phi - 1$

We prove now a general upper bound for $\alpha < \phi - 1$, but with a competitive ratio that depends on the parameter α. We split the proof, into two pieces: Theorem 5 handles the case for values of α up to $\frac{1}{2}$, . Theorem 6 contains an induction over the number of large elements in an instance, which proves the upper bound for the rest of the interval up to $\phi - 1$.

▶ **Theorem 5.** *ReserveKP has a competitive ratio of at most* $2+\alpha$ *if* $0 < \alpha \leq \frac{1}{2}$.

▶ **Theorem 6.** *ReserveKP has a competitive ratio of at most* $2+\alpha$ *if* $\frac{1}{2} < \alpha < \phi - 1$.

We consider Algorithm 2, which, unlike Algorithm 1, does not reject any offered item until it stops processing the rest of the input. In Section 5, we further discuss this class of algorithms and when they are optimal. We need the following technical lemmas.

▶ **Lemma 7.** *The size of* R *in Algorithm 2 is never larger than* $1/(2+\alpha)(1-\alpha)$.

Note, that for every considered value of α the upper bound on R is positive, that is, $(2+\alpha)(1-\alpha) \geq 0$ if α is smaller than 1. With Lemma 7 we can prove the following claim.

▶ **Lemma 8.** *If Algorithm 2 packs at least* $1/(2+\alpha)(1-\alpha)$ *into the knapsack, then its competitive ratio is at most* $2+\alpha$.

Algorithm 2 Competitive ratio $2+\alpha$ for $0 < \alpha \leq \phi - 1$.

```
R := 0;
for k = 1, ..., n do
    if x_k + (1 − α)R ≥ 1/(2 + α) then
        pack x_1, ..., x_k optimally;
        stop
    else
        reserve x_k;
        R := R + x_k
pack x_1, ..., x_n optimally
```

The next lemma allows us to restrict our attention to ordered sequences of items. It proves that, if an instance violated the upper bound for Algorithm 2, we could rearrange the first $k-1$ items in decreasing order and get another counterexample.

▶ **Lemma 9.** *If $x_1, \ldots, x_n$ is an instance for Algorithm 2, whose packing is triggered by x_k, with a competitive ratio larger than $2+\alpha$, then there is another instance $x_{i_1}, x_{i_2}, x_{i_3}, \ldots, x_{i_{k-1}}$, $x_k, \ldots, x_n$ where $(i_1, i_2, \ldots, i_{k-1})$ is a permutation of $(1, \ldots, k-1)$ and $x_{i_1} \geq x_{i_2} \geq \ldots \geq x_{i_{k-1}}$, that also has a competitive ratio larger than $2+\alpha$.*

From Lemma 8, we know that, if Algorithm 2 packs at least $1/(2+\alpha)(1-\alpha)$, this guarantees a competitive ratio of at most $2+\alpha$. Thus, if we have enough elements that are smaller than the gap of size $1 - 1/(2+\alpha)(1-\alpha)$, those elements can be packed greedily and always achieve the desired competitive ratio. Let us call *small items* those of size smaller than

$$1 - \frac{1}{(2+\alpha)(1-\alpha)} = \frac{(2+\alpha)(1-\alpha)-1}{(2+\alpha)(1-\alpha)} = \frac{1-\alpha-\alpha^2}{(2+\alpha)(1-\alpha)} \tag{2}$$

This definition is only valid when (2) is positive, that is, when $1-\alpha-\alpha^2 \geq 0$, which is the case if $0 < \alpha \leq \phi - 1$, including our desired range. We call *large items* those of larger size. Let α_4 be the unique positive real root of the polynomial $1 - 2\alpha - \alpha^2 + \alpha^3$, i. e.,

$$\alpha_4 = \frac{1}{3} + \frac{2\sqrt{7}}{3} \cos\left(\frac{1}{3} \arccos\left(-\frac{1}{2\sqrt{7}}\right) - \frac{2\pi}{3}\right) \approx 0.445 \,.$$

▶ **Lemma 10.** *Given any request sequence $x_1 \geq x_2 \geq \ldots \geq x_{k-1}, x_k, \ldots, x_n$, where x_k triggers the packing in Algorithm 2, there is at most one large item if $0 < \alpha \leq \alpha_4$ and there are at most two large items if $\alpha_4 < \alpha \leq 0.5$.*

Proof. Assume by contradiction that there exists a request sequence where $x_1 \geq x_2 \geq \ldots \geq x_i \geq \frac{1-\alpha-\alpha^2}{(2+\alpha)(1-\alpha)}$, with $k > i$. Any item x_j with $j \leq i$ satisfies

$$x_j \leq \frac{1}{2+\alpha} - (1-\alpha)(x_1 + x_2 + \ldots + x_{j-1}) \,,$$

in particular,

$$x_i \leq \frac{1}{2+\alpha} - (1-\alpha)(x_1 + x_2 + \ldots + x_{i-1}) \,. \tag{3}$$

All of the contributions of previous requests are negative, so in order to obtain a maximal value for x_i, we need that $x_1, \ldots, x_{i-1}$ are minimal, but by construction, still greater or equal than x_i. Thus, the maximal value is obtained when $x_1 = x_2 = \ldots = x_i$. In this case, we

obtain from (3) the following upper bound on the value of x_i, $x_i \leq \frac{1}{2+\alpha} - (1-\alpha)(i-1)x_i$ which we solve for x_i and obtain

$$x_i \leq \frac{1}{(2+\alpha)(i(1-\alpha)+\alpha)} \,. \tag{4}$$

We also know that x_i is a large item, thus we can also state the following lower bound on x_i

$$x_i \geq \frac{1-\alpha-\alpha^2}{(2+\alpha)(1-\alpha)} \,. \tag{5}$$

If we take into account both (4) and (5) we get $\frac{1}{(2+\alpha)(i(1-\alpha)+\alpha)} \geq \frac{1-\alpha-\alpha^2}{(2+\alpha)(1-\alpha)}$ which we solve for i to obtain

$$i \leq 1 + \frac{\alpha^2}{(1-\alpha)(1-\alpha-\alpha^2)} \,.$$

In particular, for $i = 2$ we get $2(1-\alpha)(1-\alpha-\alpha^2) \leq (1-\alpha)(1-\alpha-\alpha^2) + \alpha^2$ which is equivalent to $(1-\alpha)(1-\alpha-\alpha^2) \leq \alpha^2$ and thus to $1 - 2\alpha - \alpha^2 + \alpha^3 \leq 0$, which means that the number of large items is strictly smaller than 2 for $\alpha \leq \alpha_4$.

For $i = 3$, we get $3(1-\alpha)(1-\alpha-\alpha^2) \leq (1-\alpha)(1-\alpha-\alpha^2) + \alpha^2$ or equivalently $2(1-2\alpha+\alpha^3) \leq \alpha^2$. Hence, $2 - 4\alpha - \alpha^2 + 2\alpha^3 \leq 0$, which means that the number of large items is strictly smaller than 3 for $\alpha \leq 0.5$, since 0.5 is the unique positive real root of the left-hand-side polynomial. ◀

Now we are ready to prove the claimed competitive ratio of Algorithm 2.

Proof sketch of Theorem 5. Let us consider first the case where $\alpha \leq \alpha_4$, which means that only one large element can appear in the request sequence without triggering the packing.

We assume that there is a shortest contradictory sequence containing at least two elements, which at some point, namely with request x_k, triggers Algorithm 2 to pack. If there is no such sequence, it is rather easy to see that the claimed competitive ratio can be achieved. We consider several cases according to the sizes of the items which leads to the desired result.

In the case where $\alpha \leq 0.5$, we can use similar arguments to prove the claim. ◀

We continue with proving an upper bound for the rest of the interval, which is an induction over the number of large elements. Before we start our proof, we provide a lemma that allows us to ignore possible small elements during the proof of Theorem 6.

▶ **Lemma 11.** *Given a request sequence without small elements for which Algorithm 2 does not achieve a competitive ratio of $2+\alpha$, adding small elements to it will only improve its competitive ratio.*

Proof. Let us consider a request sequence containing only large elements $x_1 \geq \ldots \geq x_{k-1}, x_k, \ldots, x_n$, where x_k is the element triggering the packing for Algorithm 2, and the achieved competitive ratio is larger than $2+\alpha$. This means, by Lemma 8, that the packed knapsack size is smaller than $1/(2+\alpha)(1-\alpha)$. By definition, if we add enough small elements to the request sequence before x_k, the small elements can be packed greedily until the knapsack is filled up to $1/(2+\alpha)(1-\alpha)$, achieving the desired competitive ratio. If not enough of them are added before x_k, the small elements requested before x_k will be reserved but will still be able to be packed, so they will never contribute negatively to the total packing gain. ◀

Proof of Theorem 6. We prove by induction that, for any finite number of large elements before the packing, Algorithm 2 achieves the desired competitive ratio for $\alpha < \phi - 1$. The base case for zero large elements is trivial, as we can greedily reserve and later pack all small elements to get the desired competitive ratio. Let us assume that Algorithm 2 achieves the desired competitive ratio for any request sequence with less than $k-1$ large elements before the algorithm packs and stops.

If a request sequence has $k-1$ large elements we can assume by Lemma 9, that the smallest of those is x_{k-1}, and we can also assume that x_k triggers the packing by Lemma 11. We distinguish two cases.

1. When the packing is triggered, x_{k-1} is not part of an optimal packing.
   In this case, $\sum_{j \text{ is packed}} x_j + x_{k-1} > 1$. Then the following bounds hold.

$$x_{k-1} + (1-\alpha)\left(\sum_{j<k-1} x_j\right) < \frac{1}{2+\alpha}\,,$$

   $x_{k-1} \le x_1 < 1/(2+\alpha)$, and $\frac{\alpha}{1-\alpha} < 1-\alpha$ for $\alpha < \phi - 1$.
   Thus, the gain that Algorithm 2 achieves is

$$\begin{aligned}
&\sum_{j \text{ is packed}} x_j - \alpha R \\
&\quad \ge 1 - x_{k-1} - \alpha\left(\sum_{j\le k-1} x_j\right) \\
&\quad = 1 - (1+\alpha)x_{k-1} - \alpha\left(\sum_{j<k-1} x_j\right) \\
&\quad = 1 - 2\alpha x_{k-1} - (1-\alpha)\left(x_{k-1} + \frac{\alpha}{1-\alpha}\left(\sum_{j<k-1} x_j\right)\right) \\
&\quad \ge 1 - 2\alpha x_{k-1} - (1-\alpha)\left(x_{k-1} + (1-\alpha)\left(\sum_{j<k-1} x_j\right)\right) \\
&\quad \ge 1 - 2\alpha x_{k-1} - (1-\alpha)\left(\frac{1}{2+\alpha}\right) \\
&\quad \ge 1 - \frac{2\alpha}{2+\alpha} - \frac{1-\alpha}{2+\alpha} \\
&\quad \ge \frac{1}{2+\alpha}\,,
\end{aligned}$$

   as we wanted.
2. When the packing is triggered, x_{k-1} is part of an optimal packing. We consider two subcases.
   a. Taking x_{k-1} out of the request sequence still triggers the packing.
      This means that $x_k + (1-\alpha)(R - x_{k-1}) \ge \frac{1}{2+\alpha}$ holds. We can thus consider the sequence $x_1, \ldots, x_{k-2}, x_k$. This sequence has $k-2$ large elements, and x_{k-1} cannot be part of its optimal solution, thus its competitive ratio is at most $2+\alpha$ by the induction hypothesis, and the competitive ratio after adding x_{k-1} can only get better.
   b. Taking x_{k-1} out of the request sequence does not trigger the packing.

This means that $x_k + (1-\alpha)(R - x_{k-1}) < \frac{1}{2+\alpha}$, but also

$$x_k + (1-\alpha)R \geq \frac{1}{2+\alpha} ,$$

and if we let x_j be the smallest element that does not get packed, $x_j \geq x_{k-1}$ holds. Also, because of the optimality of the packing $x_k \geq x_j$, (otherwise one can take all of the reserved elements as the packing and obtain a better bound) and

$$\sum_{t \text{ is packed}} x_t - x_{k-1} + x_j > 1$$

holds. Moreover, we can bound x_j by $x_j \leq x_1 \leq \frac{1}{2+\alpha}$. With these bounds, the gain incurred by the algorithm is at least

$$\begin{aligned}
&\sum_{t \text{ is packed}} x_t - \alpha R \\
&\quad \geq 1 - x_j + x_{k-1} - \alpha R \\
&\quad \geq 1 - x_j + \frac{x_k}{1-\alpha} + R - \frac{1}{(1-\alpha)(2+\alpha)} - \alpha R \\
&\quad = 1 - x_j + \frac{x_k}{1-\alpha} + (1-\alpha)R - \frac{1}{(1-\alpha)(2+\alpha)} \\
&\quad \geq 1 - x_j + \frac{\alpha x_k}{1-\alpha} + \frac{1}{2+\alpha} - \frac{1}{(1-\alpha)(2+\alpha)} \\
&\quad = \frac{1}{2+\alpha} + \frac{1-\alpha-\alpha^2}{(1-\alpha)(2+\alpha)} - x_j + \frac{\alpha x_k}{1-\alpha} \\
&\quad \geq \frac{1}{2+\alpha} + \frac{1-\alpha-\alpha^2}{(1-\alpha)(2+\alpha)} + \frac{\alpha-(1-\alpha)}{1-\alpha} x_k \\
&\quad \geq \frac{1}{2+\alpha} + \frac{1-\alpha-\alpha^2}{(1-\alpha)(2+\alpha)} + \frac{2\alpha-1}{1-\alpha} x_k \\
&\quad \geq \frac{1}{2+\alpha} ,
\end{aligned}$$

where the last step is trivially true for any $\alpha \geq 1/2$. Thus we get the desired competitive ratio in all of the considered range for α.

This proves the induction step, and thus the desired upper bound on the competitive ratio. ◀

3.2 Lower Bound for $\sqrt{2} - 1 \leq \alpha \leq \phi - 1$

We now prove that, for the whole interval of medium reservation costs, no algorithm can achieve a better competitive ratio than $2 + \alpha$.

▶ **Theorem 12.** *Given an $\varepsilon > 0$ and an α such that $\sqrt{2}-1 \leq \alpha < 1$, there exists no algorithm for reservation knapsack achieving a competitive ratio of $2 + \alpha - \varepsilon$.*

Proof. Consider the following set of adversarial instances. First, the adversary presents an item of size $\frac{1}{2+\alpha}$. If an algorithm takes this item, the adversary presents an item of size 1, and the algorithm has a competitive ratio of $2 + \alpha$ as claimed. If an algorithm rejects this item, the adversary will present no further items; thus, the algorithm will not be competitive at all. If the item is reserved, the adversary presents a second item of size $1 - \frac{1}{2+\alpha} + \delta$, where $\delta < \frac{\varepsilon}{(2+\alpha)(2+\alpha-\varepsilon)}$. Note that the second item is larger than the first item for all $\alpha > 0$ and

that they do not fit together into the knapsack. If the item is taken, again the adversary presents an item of size 1 and ends the sequence. Thus, any algorithm reserving the first item and taking the second item has a competitive ratio of no better than

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

where the last inequality follows from the choice of δ. If the second item is rejected, the adversary will present no further items and thus any algorithm following this strategy will have a competitive ratio of

$$\frac{1-\frac{1}{2+\alpha}+\delta}{\frac{1}{2+\alpha}-\frac{\alpha}{2+\alpha}} = \frac{\frac{1+\alpha}{2+\alpha}+\delta}{\frac{1-\alpha}{2+\alpha}} > \frac{1+\alpha}{1-\alpha} > 2+\alpha$$

where the last inequality follows from the fact that $\frac{1+\alpha}{1-\alpha} > 2+\alpha$, for all values of $\alpha \geq \sqrt{2}-1$. If the second item is reserved, no further items will be presented by the adversary. Thus, any algorithm following this strategy will have a competitive ratio of

$$\begin{aligned}\frac{1-\frac{1}{2+\alpha}+\delta}{1-\frac{1}{2+\alpha}+\delta-\frac{\alpha}{2+\alpha}-\alpha+\frac{\alpha}{2+\alpha}-\alpha\delta} &= \frac{\frac{1+\alpha}{2+\alpha}+\delta}{\frac{1-\alpha-\alpha^2}{2+\alpha}+\delta(1-\alpha)} \\ &> \frac{\frac{1+\alpha}{2+\alpha}}{\frac{1-\alpha-\alpha^2}{2+\alpha}+\delta} > \frac{1-\alpha}{\frac{1-\alpha-\alpha^2}{2+\alpha}+\delta},\end{aligned}$$

where the last inequality again follows from the fact that $\frac{1+\alpha}{1-\alpha} > 2+\alpha$, for all values of $\alpha \geq \sqrt{2}-1$. We claim that

$$\frac{1-\alpha}{\frac{1-\alpha-\alpha^2}{2+\alpha}+\delta} > 2+\alpha\,.$$

Since $1-\alpha-\alpha^2$ is positive for all $\alpha < \phi - 1$, this is equivalent to $\frac{1-\alpha}{2+\alpha} > \frac{1-\alpha-\alpha^2}{2+\alpha}+\delta$ or equivalently $\alpha^2 > (2+\alpha)\delta$ which is true for all reasonable choices of δ. ◀

4 High Reservation Costs

In this section, we analyze the competitive ratio in the remaining interval of reservation costs between $\phi - 1$ and 1. We prove a tight bound of $\frac{1}{1-\alpha}$ on the competitive ratio.

4.1 Upper Bound for $\phi - 1 \leq \alpha < 1$

For proving an upper bound, we consider Algorithm 3 and first bound its reservation costs.

▶ **Lemma 13.** *For Algorithm 3, the reservation cost R is never larger than* 1.

We are now ready to prove the desired competitive ratio for Algorithm 3.

▶ **Theorem 14.** *Algorithm 3 is an online algorithm for* ReserveKP *achieving a competitive ratio of at most $\frac{1}{1-\alpha}$, for all $\phi - 1 \leq \alpha < 1$.*

Proof sketch. The full proof of this theorem is structurally very similar to that of Theorem 6. Let us first assume that we run Algorithm 3 on an instance $x_1, \ldots, x_n$, and no element triggers the packing. This means that all elements are reserved. But we know by Lemma 13

Algorithm 3 Algorithm for $\phi - 1 \leq \alpha < 1$.

```
R := 0;
for k = 1, ..., n do
    if x_k + (1 − α)R ≥ 1 − α then
        pack x_1, ..., x_k optimally;
        stop
    else
        reserve x_k;
        R := R + x_k
pack x_1, ..., x_n optimally
```

that the reservation never exceeds the capacity of the knapsack. This means that the optimal solution packs all offered elements. Thus, the algorithm achieves a competitive ratio of

$$\frac{\sum_{i=1}^{n} x_{k-1}}{(\sum_{i=1}^{n} x_{k-1}) - \alpha \sum_{i=1}^{n} x_{k-1}} = \frac{1}{1-\alpha} .$$

Now, it remains to analyze the case where an instance $x_1, \ldots, x_n$ triggers the packing, for some x_k. We do an induction on the value of k. If $k = 1$, the first item offered triggers the packing, thus it holds that $x_1 \geq 1 - \alpha$ and the gain of the algorithm is at least $1 - \alpha$ as we expected. Now, we assume that Algorithm 3 has a gain of at least $1 - \alpha$ on any request sequence triggering the algorithm to pack and stop before k elements are offered.

We proceed similarly to the proof of Theorem 6, making a case distinction over whether x_{k-1} is part of an optimal packing. ◀

4.2 Lower Bound for $\phi - 1 \leq \alpha < 1$

Now we present a lower bound of $\frac{1}{1-\alpha}$ for $\phi - 1 \leq \alpha < 1$. Observe that this lower bound works for all α. However, for smaller values of α, we see that $2 + \alpha > \frac{1}{1-\alpha}$, so Theorem 12 already gives a better lower bound for that range.

▶ **Theorem 15.** *For any $\varepsilon > 0$ and any α such that $\phi - 1 \leq \alpha < 1$, no online algorithm for* ReserveKP *can achieve a competitive ratio of $\frac{1}{1-\alpha} - \varepsilon$.*

Proof sketch. The proof of this theorem is structurally similar to that of Theorem 12. ◀

5 Nonrejecting Algorithms

We call an algorithm *nonrejecting* if it only chooses from one of the options *pack*, *reserve* or *stop and pack* for each item that it is given. Such an algorithm is thus unable to reject any items until the point where it discards the remaining elements of an instance.

A valid intuition for ReserveKP might be the following: If the cost of reservation is very small, rejecting an item should not be necessary, as even when an item cannot be packed, the cost of reserving it is negligible. On the other hand, when the reservation cost is rising, aggressively reserving items may seem like a very bad strategy, as the risk of not being able to utilize reserved items may come to mind. Interestingly, both of these intuitions turn out to be wrong, which we will show by first giving a lower bound for nonrejecting algorithms that exceeds the upper bound of a rejecting algorithm for small α and that tightly matches the upper bound of a nonrejecting algorithm for bigger α.

We first provide a lower bound on the competitive ratio for nonrejecting algorithms.

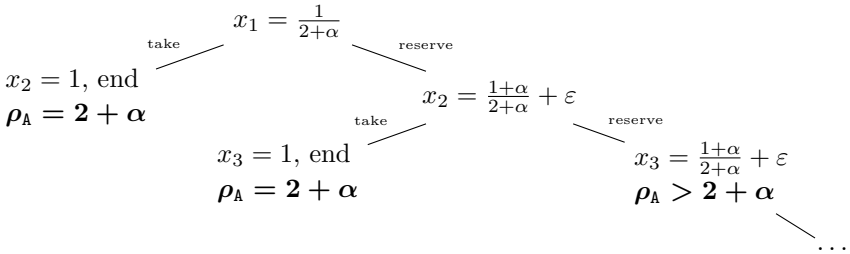

Figure 4 Sketch of the adversarial strategy that is used in the proof of Theorem 16. Upon continued reservation, a new item of the same size is repeatedly presented.

▶ **Theorem 16.** *There exists no deterministic online algorithm for* ReserveKP *that does not reject any elements with a competitive ratio better than* $2 + \alpha$ *for any* $0 < \alpha < 1$.

Proof sketch. The proof of this theorem is structurally similar to the proofs of Theorems 12 and 15. A sketch of the adversarial strategy can be found in Figure 4. ◀

Combined with the upper bound given in Lemma 2 we see that an algorithm that is unable to reject items performs quite a bit worse than one that is able to reject items, such as Algorithm 1. Thus, an algorithm needs to be able to reject items to become 2-competitive for small values of α. On the other hand, the lower bound provided in Theorem 16 matches the upper bound of Theorem 5, which is based on the nonrejecting Algorithm 2. Thus, there are nonrejecting algorithms for every $\alpha \geq \sqrt{2} - 1$ that are at least as good as any other algorithms that are able to reject items.

6 Further Work

In this work, we give first bounds on this new model, but left a gap between the upper and lower bounds for $\frac{1}{6} < \alpha < \sqrt{2} - 1$. It would be of interest to us how the complete picture of all tight bounds looks like. We left the case open where the item costs are not proportional to their sizes, which seems also constantly competitive depending on α. We also leave the randomized and advice complexity of ReserveKP open.

Furthermore, one could consider a variant where reservation costs are refunded if the item is used or where one pays a fee proportional to the size of all reserved items in each step.

The concept of reservation may be applied to other online problems such as online call admission problems in networks [2, 7, 19] or problems of embedding guest graphs into a host graph. In the online path packing problem, one packs paths in a edge-disjoint way (sometimes node-disjointly) into a graph, which is a generalization of ReserveKP, thus inheriting all lower bounds. The offline version was studied on many types of graphs, with an incomplete selection being [10, 21, 23, 8].

References

1 Susanne Albers. Energy-efficient algorithms. *Commun. ACM*, 53(5):86–96, 2010. doi:10.1145/1735223.1735245.

2 Baruch Awerbuch, Yossi Azar, and Serge A. Plotkin. Throughput-competitive on-line routing. In *34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993*, pages 32–40. IEEE Computer Society, 1993. doi:10.1109/SFCS.1993.366884.

3 Hans-Joachim Böckenhauer, Elisabet Burjons, Juraj Hromkovič, Henri Lotze, and Peter Rossmanith. Online simple knapsack with reservation costs. *CoRR*, abs/2009.14043, 2020. arXiv:2009.14043.

4 Hans-Joachim Böckenhauer, Richard Dobson, Sacha Krug, and Kathleen Steinhöfel. On energy-efficient computations with advice. In Dachuan Xu, Donglei Du, and Ding-Zhu Du, editors, *Computing and Combinatorics - 21st International Conference, COCOON 2015, Beijing, China, August 4-6, 2015, Proceedings*, volume 9198 of *Lecture Notes in Computer Science*, pages 747–758. Springer, 2015. doi:10.1007/978-3-319-21398-9_58.

5 Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Královič, Richard Královič, and Tobias Mömke. Online algorithms with advice: The tape model. *Inf. Comput.*, 254:59–83, 2017. doi:10.1016/j.ic.2017.03.001.

6 Hans-Joachim Böckenhauer, Dennis Komm, Richard Královič, and Peter Rossmanith. The online knapsack problem: Advice and randomization. *Theor. Comput. Sci.*, 527:61–72, 2014. doi:10.1016/j.tcs.2014.01.027.

7 Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

8 Darryn E. Bryant. Packing paths in complete graphs. *J. Comb. Theory, Ser. B*, 100(2):206–215, 2010. doi:10.1016/j.jctb.2009.08.004.

9 Stefan Dobrev, Rastislav Královič, and Dana Pardubská. Measuring the problem-relevant information in input. *ITA*, 43(3):585–613, 2009. doi:10.1051/ita/2009012.

10 András Frank. Packing paths in planar graphs. *Combinatorica*, 10(4):325–331, 1990. doi:10.1007/BF02128668.

11 Xin Han, Yasushi Kawase, and Kazuhisa Makino. Online unweighted knapsack problem with removal cost. *Algorithmica*, 70(1):76–91, 2014. doi:10.1007/s00453-013-9822-z.

12 Xin Han, Yasushi Kawase, Kazuhisa Makino, and Haruki Yokomaku. Online knapsack problems with a resource buffer. In Pinyan Lu and Guochuan Zhang, editors, *30th International Symposium on Algorithms and Computation, ISAAC 2019, December 8-11, 2019, Shanghai University of Finance and Economics, Shanghai, China*, volume 149 of *LIPIcs*, pages 28:1–28:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ISAAC.2019.28.

13 Juraj Hromkovič, Rastislav Královič, and Richard Královič. Information complexity of online problems. In Petr Hliněný and Antonín Kučera, editors, *Mathematical Foundations of Computer Science 2010, 35th International Symposium, MFCS 2010, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6281 of *Lecture Notes in Computer Science*, pages 24–36. Springer, 2010. doi:10.1007/978-3-642-15155-2_3.

14 Oscar H. Ibarra and Chul E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM*, 22(4):463–468, 1975. doi:10.1145/321906.321909.

15 Sandy Irani, Sandeep K. Shukla, and Rajesh Gupta. Algorithms for power savings. *ACM Trans. Algorithms*, 3(4):41, 2007. doi:10.1145/1290672.1290678.

16 Kazuo Iwama and Shiro Taketomi. Removable online knapsack problems. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan J. Eidenbenz, and Ricardo Conejo, editors, *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings*, volume 2380 of *Lecture Notes in Computer Science*, pages 293–305. Springer, 2002. doi:10.1007/3-540-45465-9_26.

17 Kazuo Iwama and Guochuan Zhang. Online knapsack with resource augmentation. *Inf. Process. Lett.*, 110(22):1016–1020, 2010. doi:10.1016/j.ipl.2010.08.013.

18 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.

19 Dennis Komm. *An Introduction to Online Computation - Determinism, Randomization, Advice*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2016. doi:10.1007/978-3-319-42749-2.

20 Alberto Marchetti-Spaccamela and Carlo Vercellis. Stochastic on-line knapsack problems. *Math. Program.*, 68:73–104, 1995. doi:10.1007/BF01585758.

21 Alexander Schrijver and Paul D. Seymour. Packing odd paths. *J. Comb. Theory, Ser. B*, 62(2):280–288, 1994. doi:10.1006/jctb.1994.1070.

22 Daniel Dominic Sleator and Robert Endre Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, 1985. doi:10.1145/2786.2793.

23 Natalia Vanetik. Path packing and a related optimization problem. *J. Comb. Optim.*, 17(2):192–205, 2009. doi:10.1007/s10878-007-9107-z.