Math Meth Oper Res (2016) 83:207–242
DOI 10.1007/s00186-015-0526-9

ORIGINAL ARTICLE

The online knapsack problem with incremental capacity

Clemens Thielen[1] · Morten Tiedemann[2] · Stephan Westphal[3]

Received: 22 January 2015 / Accepted: 3 December 2015 / Published online: 30 December 2015

Abstract We consider an online knapsack problem with incremental capacity. In each time period, a set of items, each with a specific weight and value, is revealed and, without knowledge of future items, it has to be decided which of these items to accept. Additionally, the knapsack capacity is not fully available from the start but increases by a constant amount in each time period. The goal is to maximize the overall value of the accepted items. This setting extends the basic online knapsack problem by introducing a dynamic instead of a static knapsack capacity and is applicable to classic problems such as resource allocation or one-way trading. In contrast to the basic online knapsack problem, for which no competitive algorithms exist, the setting of incremental capacity facilitates the development of competitive algorithms for a bounded time horizon. We provide a competitive analysis of deterministic and randomized online algorithms for the online knapsack problem with incremental capacity and present lower bounds on

Morten Tiedemann is supported by the German Research Foundation (DFG), Grant GRK 1703/1 for the RTG "Resource Efficiency in Interorganizational Networks".

✉ Morten Tiedemann
m.tiedemann@math.uni-goettingen.de

Clemens Thielen
thielen@mathematik.uni-kl.de

Stephan Westphal
stephan.westphal@tu-clausthal.de

[1] Department of Mathematics, University of Kaiserslautern, Paul-Ehrlich-Str. 14, 67663 Kaiserslautern, Germany

[2] DFG RTG 1703, Institute for Numerical and Applied Mathematics, University of Göttingen, Lotzestr. 16-18, 37083 Göttingen, Germany

[3] Institute for Applied Stochastics and Operations Research, Clausthal University of Technology, Erzstraße 1, 38647 Clausthal-Zellerfeld, Germany

the competitive ratio achievable by online algorithms for the problem. Most of these lower bounds match the competitive ratios achieved by our online algorithms exactly or differ only by a constant factor.

Keywords Online optimization · Competitive analysis · Knapsack problems · Dynamic capacity

1 Introduction

In this paper, we consider an online version of the knapsack problem in which the capacity of the knapsack increases stepwise over a given number of discrete time periods. In each time period, a set of requests (items), each with a specific weight and value, is revealed and, without knowledge of future requests, it has to be decided which of these requests to accept (i.e., to pack into the knapsack). Furthermore, the knapsack capacity changes dynamically over time, i.e., the capacity is not fully available from the start but a constant additional amount of capacity becomes available in each time period. Hence, if $k \geq 1$ denotes the amount of additional capacity that becomes available in each time period, the available capacity in time period $i \in \{1, \ldots, T\}$ is $k \cdot i$ minus the total weight of all requests that have been accepted in time periods 1 to $i - 1$, where T denotes the total number of time periods considered. The goal is to maximize the overall value of the accepted requests while respecting the capacity constraint in each time period.

The initial idea for this work stems from an interdisciplinary project with forestry scientists concerning the problem of resource allocation in the context of renewable resources. Consider a timber producer who frequently receives supply from the forest and has to deal with requests offering specific prices for different amounts of timber in each time period. Broadly speaking: in each time period, offerings with specific prices for certain amounts of the resource are revealed and, at the same time, additional resources become available. Which offers should be accepted in order to maximize the total profit? An approach to answering this question is given by the model presented and analyzed in this work.

Another motivation for studying online knapsack problems with incremental capacity is the generalization of the classic one-way trading problem introduced in El-Yaniv et al. (1992) to dynamically increasing funds. In the classic one-way trading problem, an online player is given an initial amount of dollars that should be converted to yen over a given number of days. Each day, a new exchange rate is announced and the player has to decide how many dollars to convert in order to maximize the total amount of yen obtained after the last day. This can be seen as a special case of the online knapsack problem by viewing the initial amount of dollars as the knapsack capacity and introducing a request for each possible amount that could be traded on each day, where all requests on a specific day have the same value to weight ratio (which corresponds to the exchange rate on this day). Considering the online knapsack problem with incremental capacity then corresponds to receiving a certain amount of dollars for conversion every day instead of having the total amount of dollars available already on the first day.

Both applications above motivate a thorough investigation of the described setting. In the following, we discuss the setting of the online knapsack problem with incremental capacity, present deterministic and randomized online algorithms for the problem, and derive lower bounds on the competitive ratio achievable by online algorithms.

The *competitive ratio* is the standard measure for the quality of online algorithms in *competitive analysis* (cf. Borodin and El-Yaniv 1998), where, for each request sequence σ, the value $\text{ALG}(\sigma)$ obtained by an online algorithm ALG is compared to the optimal value $\text{OPT}(\sigma)$ achievable on that sequence. For maximization problems as studied in this paper, the competitiveness of deterministic and randomized algorithms is defined as stated in the following definitions:

Definition 1 (*Competitiveness of a deterministic algorithm*) A deterministic online algorithm ALG is called *c-competitive* for a constant $c \geq 1$ if $\text{ALG}(\sigma) \geq \frac{1}{c} \cdot \text{OPT}(\sigma)$ for all request sequences σ.

A *randomized online algorithm* is a probability distribution over deterministic algorithms. We measure the solution quality obtained by a randomized online algorithm by considering its competitive ratio against an *oblivious adversary*, who must construct the input sequence in advance based only on the description of the online algorithm but before any moves are made (cf. Borodin and El-Yaniv 1998):

Definition 2 (*Competitiveness of a randomized algorithm*) A randomized online algorithm ALG, distributed over a set $\{\text{ALG}_y\}$ of deterministic algorithms, is *c-competitive (against an oblivious adversary)* for a constant $c \geq 1$ if

$$\mathbb{E}_Y[\text{ALG}_y(\sigma)] \geq \frac{1}{c} \cdot \text{OPT}(\sigma)$$

for all request sequences σ. Here, $\mathbb{E}_Y[\cdot]$ denotes the expectation with respect to the probability distribution Y over $\{\text{ALG}_y\}$ that defines ALG.

The *competitive ratio* of a deterministic or randomized online algorithm is defined as the infimum over all c such that the algorithm is c-competitive.

1.1 Previous work

The offline version of the knapsack problem and a wide range of its variants have been studied for many years and are covered in the literature comprehensively. For a full-scale presentation of methods and techniques available for the solution of the knapsack problem, we refer to the textbooks (Martello and Toth 1990; Kellerer et al. 2004). The classic 0/1 knapsack problem is $\mathcal{NP}$-hard (Karp 1972), but it admits an FPTAS as first shown by Ibarra and Kim (1975). A constant-time randomized approximation algorithm based on weighted sampling was proposed in Ito et al. (2012).

An offline knapsack problem with incremental capacity was considered by Bienstock et al. (2013). Here, all items are available already in the first time period, but the knapsack capacity grows weakly as a function of time throughout the time periods $t = 1, \ldots, T$. Moreover, in the model studied in Bienstock et al. (2013), the value obtained from packing an item may depend on the time period t in which the item is packed via an item-independent discounting factor $\Delta_t > 0$. For this general offline model, they presented a constant-factor approximation algorithm under the assumption that the capacity function of the knapsack is upper bounded by a polynomial function of t. This improves upon the previously best approximation algorithm for the problem, which was a general purpose approximation algorithm for incremental optimization problems due to Hartline and Sharp (2006) that obtained an approximation ratio of $\mathcal{O}(\log T)$. For the case that $\Delta_t = 1$ for all t and $T = \mathcal{O}(\log n)$, Bienstock et al. (2013) presented a polynomial-time approximation scheme (PTAS) for the problem (where n denotes the number of items).

A variant of the knapsack problem in which all items are given from the beginning, but the knapsack capacity is unknown was recently studied by Disser et al. (2014). Here, whenever an algorithm attempts to pack an item that does not fit into the knapsack, the item is discarded. If the item fits, it has to be included in the packing. For this setting, a policy was given that packs a total value within a factor 2 of the optimum packing, irrespective of the actual capacity. For the case that the value of each item equals its weight, this factor can be improved to the golden ratio $(1+\sqrt{5})/2 \approx 1.618$. If the packing stops once the algorithm tries to pack an item that does not fit into the remaining knapsack capacity, there exists no policy that obtains a constant fraction of the optimal value for every possible capacity, but Megow and Mestre (2013) provided an algorithm that constructs for each instance a solution that comes arbitrarily close to the best possible worst-case factor.

The online knapsack problem was introduced by Marchetti-Spaccamela and Vercellis (1995), who showed that the general online knapsack problem does not admit any competitive online algorithms. Consequently, researchers began to study the online knapsack problem under additional assumptions that allow the design of competitive algorithms or in stochastic settings using average case analysis.

A stochastic version of the online knapsack problem where the values and the weights are independent and identically distributed random variables was studied by Marchetti-Spaccamela and Vercellis (1995). For this setting, they obtained a linear-time algorithm for which the obtained objective value differs from the optimum by a factor of $\mathcal{O}(\log^{3/2} n)$ in expectation, where n denotes the number of items. This factor was subsequently improved to $\mathcal{O}(\log n)$ by Lueker (1998). Further generalizations of the stochastic online knapsack problem were studied, e.g., in Papastavrou et al. (1996), Kleywegt and Papastavrou (1998), Slyke and Young (2000).

Babaioff et al. (2007) considered the online knapsack problem without any assumptions regarding the distribution of weights, but made the assumption that elements arrive in a random order. For this setting, they presented a $10e$-competitive algorithm for the case of general weights and two e-competitive algorithms for the case of unit weights as the number of items tends to infinity.

Zhou et al. (2008) applied competitive analysis to the online knapsack problem under the assumption that the value-to-weight ratio of every item is within a known range $[L, U]$ and each weight is assumed to be much smaller than the knapsack capacity. They presented a $\log(U/L) + 1$-competitive deterministic online algorithm and showed a matching lower bound on the competitive ratio even for randomized online algorithms. In particular, this shows that this variant of the online knapsack problem does not admit any online algorithms with a constant competitive ratio.

Further variants of the online knapsack problem analyzed using competitive analysis include settings with removable items, which means that accepted items can be removed to give way for newly arriving items. For the case that the value of each item equals its weight, this variant was considered in Iwama and Taketomi (2002), where a deterministic $(1+\sqrt{5})/2 \approx 1.618$-competitive algorithm and a $(1+\sqrt{5})/2 - \epsilon$ lower bound on the competitive ratio of any deterministic algorithm were presented. Recently, Han et al. (2015) presented a randomized $10/7$-competitive algorithm and a lower bound of $5/4$ on the competitive ratio of any randomized online algorithm for this case. For the case of general value-to-weight ratios, no deterministic competitive algorithms exist (Iwama and Zhang 2003) and no randomized online algorithm can achieve a competitive ratio smaller than $1 + 1/e$ (Han et al. 2015), but a randomized 2-competitive algorithm is known (Han et al. 2015).

Another possibility to circumvent the nonexistence of competitive algorithms for the online knapsack problem is the approach of resource augmentation. In this setting, the online player is allowed to use a knapsack of capacity $R \geq 1$ while the adversary uses a knapsack of capacity one. In Iwama and Zhang (2007), this approach was combined with the possibility of removing already accepted items and it was shown that the standard greedy algorithm achieves a competitive ratio of $1/(R-1)$ for each $1 < R \leq 2$, and this is best possible for a deterministic online algorithm. If the value of each item equals its weight, they achieve an optimal competitive ratio of $2/(2R-1)$ for $R \geq (1+\sqrt{2})/2$ and of $(\sqrt{4R+1}+1)/(2R)$ for $R < (1+\sqrt{2})/2$. In Noga and Sarbua (2005), the online partially fractional knapsack problem was studied using resource augmentation and the possibility of removing already accepted items. Using a knapsack of capacity $1 \leq R \leq 2$, this allowed the design of a deterministic $2/R$-competitive algorithm and this is best possible for deterministic algorithms. Moreover, a randomized $(2+R)/(2R)$-competitive online algorithm was presented.

1.2 Our contribution

We consider the model for the online knapsack problem with incremental capacity described in the introduction. For the case of unit weight requests and unit incremental capacity (i.e., one additional unit of capacity becoming available in each time period), we give a deterministic T-competitive online algorithm and a matching lower bound on the competitive ratio of any deterministic online algorithm, where T is the given number of time periods. For unit weights and k-incremental capacity (where $k \geq 2$ additional units of capacity become available in each time period), a deterministic $(T+1)k/(2k-1)$-competitive algorithm is proposed. This competitive ratio approaches the lower bound on the competitive ratio of any randomized (and deterministic) algorithm for $k \to \infty$. Moreover, a randomized algorithm with a competitive ratio of $(T+1)/2$ is developed for this setting, matching the lower bound for any randomized algorithm.

For the case that general nonnegative weights are allowed, we show that no competitive online algorithm exists for the problem. However, for limited weights in $\{1, \ldots, k\}$ and k-incremental capacity, we present a competitive deterministic online algorithm

Table 1 Summary of our results: deterministic and randomized lower and upper bounds

Item type	Unit weights		Limited weights		Removable[a]	
Incremental cap.	$k = 1$	$k \geq 2$	$k = 1$	$k \geq 2$	$k = 1$	$k \geq 2$
det.						
LB	T	$\frac{T+1}{2}$	T	$\left\lfloor \frac{Tk}{\left\lfloor \frac{k}{2} \right\rfloor + 1} \right\rfloor$	1	$\sqrt{2}$
UB	T	$\frac{(T+1)k}{2k-1}$	T	$2T - 1$	1	3
rand.						
LB	$\frac{T+1}{2}$	$\frac{T+1}{2}$	$\frac{T+1}{2}$	$\frac{T+1}{2}$	–	–
UB	$\frac{T+1}{2}$	$\frac{T+1}{2}$	$\frac{T+1}{2}$	$3\left(\frac{T+1}{2}\right)$	–	–

[a] OKIC with removable items and limited weights

and a lower bound on the competitive ratio of any deterministic online algorithm that approaches the competitive ratio of the proposed algorithm for $k \to \infty$.[1] For the randomized case, we present a $3\,((T+1)/2)$-competitive online algorithm matching the lower bound of $(T+1)/2$ up to a factor of 3.

In order to achieve algorithms with competitive ratios independent of T, we study the setting of removable items, i.e., the online player is entitled to remove previously accepted items from the knapsack in any time period, and present a 3-competitive deterministic algorithm and a lower bound of $\sqrt{2}$ on the competitive ratio of any deterministic online algorithm for the case of limited weights in $\{1, \dots, k\}$. We remark that, even though we assume limited weights, we allow arbitrary value-to-weight ratios, so the existence of a deterministic online algorithm with constant competitive ratio for the setting of removable items is in sharp contrast to the classical online knapsack problem with removable items, which does not admit any deterministic competitive online algorithms if arbitrary value-to-weight ratios are allowed (Iwama and Zhang 2003). Our results are summarized in Table 1.

In addition to these results for the single knapsack case, we show that all our algorithms can be extended generically to multiple knapsacks while only increasing their competitiveness by one.

We remark that, even though we assume that the total number T of time periods is known from the beginning, most of our deterministic online algorithms do not make use of this knowledge, so they obtain the same competitiveness in a time-oblivious setting where T is *not* known to the online algorithm. Hence, all our deterministic upper bounds except for the $(T+1)k/2k-1$ upper bound for unit weights and $k \geq 2$ transfer directly to the time-oblivious setting.

The rest of the paper is structured as follows: in Sect. 2, we formally introduce the online knapsack problem with incremental capacity. In Sect. 3, we derive lower bounds on the competitive ratio achievable by both deterministic and randomized

[1] Note that, for unit incremental capacity $k = 1$, the case of limited weights in $\{1, \dots, k\}$ coincides with the unit weight case, so the results shown for unit weights carry over to the limited weight setting if $k = 1$.

online algorithms for the case of unit weight requests. Deterministic and randomized competitive online algorithms for this case are presented in Sect. 4. In Sect. 5, the cases of general weights as well as limited weights in $\{1, \ldots, k\}$ are considered. In Sect. 6, the setting of removable items is analyzed. Finally, we consider multiple knapsacks in Sects. 7 and 8 contains a summary of our contribution and directions for future research.

2 Problem definition

In this section, we formally introduce the online knapsack problem with incremental capacity.

We consider a time horizon $T \in \mathbb{N}^+$ and N requests $r_i = (d_i, v_i, w_i)$, each consisting of a time period $d_i \in \{1, \ldots, T\}$ in which the request is offered, a value $v_i \in \mathbb{R}^+$, and a weight $w_i \in \mathbb{N}^+$.

For the sake of simplicity, we first consider unit weights, i.e., $w_i \equiv 1$. The case of general weights is considered in Sect. 5. The time horizon T is known to an online algorithm, whereas the number N of requests is not. In each time period $t \in \{1, \ldots, T\}$, the requests r_i with $d_i = t$ are revealed and an online algorithm has to decide which of these requests to accept. The requests with $d_i = t$ that are not accepted in time period t are lost.[2] The knapsack capacity is increased by a constant amount of $k \in \mathbb{N}^+$ units in each time period, where k is known to an online algorithm. Denoting the available knapsack capacity in time period t by c_t, this means that $c_1 = k$ and $c_t = c_{t-1} + k - |S_{t-1}|$ for $t \geq 2$, where S_{t-1} denotes the set of indices of requests accepted by the online algorithm at time $t - 1$. The objective is to maximize the total value of accepted requests over all time periods $1, \ldots, T$ while not accepting requests of total size larger than c_t in any time period t.

The problem described above is in the following referred to as the *online knapsack problem with incremental capacity* (OKIC).

3 Lower bounds

Before we discuss competitive algorithms for OKIC in Sect. 4, lower bounds on the competitive ratio of any deterministic and randomized online algorithm are given in this section.

3.1 A lower bound for deterministic online algorithms

For a lower bound on deterministic algorithms for OKIC, we first consider the case $k = 1$.

[2] Note that, even if one would consider requests that remain valid for several time periods, it would not be advantageous for the adversary to reveal requests that remain valid for more than one period. Therefore, this possibility is not considered in our model.

Theorem 1 *For $k = 1$, no deterministic online algorithm for* OKIC *can achieve a competitive ratio smaller than T.*

Proof Consider the following requests: in each time period $t' = 1, \dots, T$, there are t' identical requests $r_{t'_1}, \dots, r_{t'_{t'}}$ with

$$r_{t'_i} = (t', v^{t'}, 1), \quad v \geq 1, \quad i = 1, \dots, t'.$$

For each $t \in \{1, \dots, T\}$, consider the request sequence σ_t in which the adversary presents the requests $r_{t'_1}, \dots, r_{t'_{t'}}$ in the time periods $t' = 1, \dots, t$ and no further requests in the time periods $t' > t$.

First, we show by induction on t that, when choosing v large enough, every deterministic online algorithm ALG must accept exactly one request in each time period $t' = 1, \dots, t$ in order to be competitive on the request sequence σ_t: for $t = 1$, ALG must clearly accept the only available request r_{1_1} in order to be competitive. Now consider σ_t for $t \geq 2$ and assume that the statement holds for all $t' < t$. Note that there are only two options for ALG in time period t: It can either accept one request or no request at all. This follows since, by induction hypothesis, ALG accepted exactly one request in each time period $t' < t$, so, since there is only one additional unit of capacity in each time period, the available capacity of ALG at time t equals one.

If ALG accepts no request in time period t, the adversary accepts all available requests $r_{t_1}, \dots, r_{t_t}$ in time period t (and no requests in earlier time periods). For the competitive ratio, this yields

$$\frac{\text{OPT}}{\text{ALG}} = \frac{t v^t}{\sum_{j=1}^{t-1} v^j} \geq t v^t \cdot \frac{v-1}{v^t - 1} = t \cdot \frac{v-1}{1 - \frac{1}{v^t}} \to \infty \quad \text{for } v \to \infty.$$

Consequently, on the request sequence σ_T obtained for $t = T$, any competitive deterministic online algorithm for OKIC accepts exactly one request r_{t_i} in each time period $t' \leq T$ if v is large enough. The optimal offline algorithm, on the other hand, accepts no requests until time T and then accepts all the requests $r_{T_1}, \dots, r_{T_T}$.
Analogous to the calculation above, this leads to a competitive ratio of

$$\frac{\text{OPT}}{\text{ALG}} = \frac{T v^T}{\sum_{j=1}^{T} v^j} \geq T v^T \cdot \frac{v-1}{v^{T+1} - 1} = T \cdot \frac{1 - \frac{1}{v}}{1 - \frac{1}{v^{T+1}}} \to T \quad \text{for } v \to \infty.$$

□

For $k \geq 2$, we show a lower bound of $(T+1)/2$ on the competitive ratio even for randomized online algorithms in the following subsection. This lower bound is asymptotically best possible even for deterministic algorithms since we present a deterministic online algorithm whose competitive ratio approaches the lower bound asymptotically for $k \to \infty$ in Sect. 4.2.

3.2 A lower bound for randomized online algorithms

We now present a lower bound on the competitive ratio of any randomized online algorithm for OKIC.

Theorem 2 *For $T \geq 2$ and $k \in \mathbb{N}^+$, no randomized online algorithm for* OKIC *can achieve a competitive ratio smaller than* $(T+1)/2$.

Proof For each $i \in \{1, \dots, T\}$, consider the request sequence σ_i consisting of $j \cdot k$ requests with value v^j in each time period $j = 1, \dots, i$, i.e., for each $j \in \{1, \dots, i\}$, we have $j \cdot k$ requests of the form $r_j = (j, v^j, 1)$ for some $v > 0$. The optimal solution for sequence σ_i is obviously to save up capacity until time period i and then accept all requests of value v^i, resulting in $\text{OPT}(\sigma_i) = i \cdot k \cdot v^i$.

With respect to the request sequences described above, each deterministic online algorithm ALG is characterized by the number of requests accepted at time j, which we denote by α_j. Using this notation, the profit ratio of any deterministic online algorithm ALG and the optimal offline algorithm OPT with respect to request sequence σ_i is given by

$$\frac{\text{ALG}(\sigma_i)}{\text{OPT}(\sigma_i)} = \frac{\sum_{j=1}^{i} \alpha_j v^j}{ikv^i}.$$

In the following, we derive a probability distribution p over the request sequences σ_i such that

$$\mathbb{E}_p\left[\frac{\sum_{j=1}^{i} \alpha_j v^j}{ikv^i}\right] \leq \frac{2}{T+1},$$

for every deterministic online algorithm ALG. The claimed lower bound on the competitive ratio of any randomized online algorithm then follows from Yao's principle (cf. Borodin and El-Yaniv 1998; Yao 1977).

For a given probability distribution p, we have

$$\mathbb{E}_p\left[\frac{\text{ALG}(\sigma_i)}{\text{OPT}(\sigma_i)}\right] = \sum_{i=1}^{T} p_i \frac{\sum_{j=1}^{i} \alpha_j v^j}{ikv^i} = \sum_{i=1}^{T}\sum_{j=1}^{i} \frac{p_i v^{j-i}}{ik}\alpha_j = \sum_{j=1}^{T}\sum_{i=j}^{T} \frac{p_i v^{j-i}}{ik}\alpha_j,$$

where p_i denotes the probability that request sequence σ_i occurs. The sum of accepted requests up to time period i is at most ik, since there are k additional units of capacity in each time period, i.e., $\sum_{j=1}^{i} \alpha_j \leq ik$ for $i = 1, \dots, T$.

Therefore, the maximum profit ratio of any deterministic online algorithm and the optimal offline algorithm can be obtained by the following integer program in the variables α_j:

$$\max \quad \sum_{j=1}^{T}\sum_{i=j}^{T} \frac{p_i v^{j-i}}{ik}\alpha_j \tag{P}$$

$$\begin{aligned}\text{s.t.} \quad & \sum_{j=1}^{i} \alpha_j \le ik \qquad && \text{for } i = 1, \dots, T \\ & \alpha_j \in \mathbb{N}^+ && \text{for } j = 1, \dots, T.\end{aligned}$$

In order to determine an upper bound on (P), it is sufficient to find a feasible solution to the dual of the linear relaxation of (P), which is given by

$$\min \quad \sum_{i=1}^{T} ik\Pi_i \tag{D}$$

$$\text{s.t.} \quad \sum_{i=j}^{T} \Pi_i \ge \sum_{i=j}^{T} \frac{p_i v^{j-i}}{ik} \qquad \text{for } j = 1, \dots, T \tag{1}$$

$$\Pi_i \ge 0 \qquad \text{for } i = 1, \dots, T.$$

We set

$$p_1 = \frac{2}{T(T+1)} \tag{2}$$

and

$$p_i = ip_1 \quad \text{for } i = 2, \dots, T. \tag{3}$$

This is at least a feasible choice for the p_i, $i = 1, \dots, T$, since $p_i \ge 0$ and

$$\sum_{i=1}^{T} p_i \overset{(3)}{=} \sum_{i=1}^{T} ip_1 = p_1 \frac{T(T+1)}{2} \overset{(2)}{=} 1.$$

In the following, we show that, by this choice of the variables p_i, a feasible solution of the program (D) with objective value $2/(T+1)$ can be found.

For this purpose, we replace the inequality constraints (1) in (D) by equality constraints. For the variables Π_j, $j = 1, \dots, T$, we then obtain

$$\begin{aligned}\Pi_j &\overset{(1)}{=} \sum_{i=j}^{T} \frac{p_i v^{j-i}}{ik} - \sum_{i=j+1}^{T} \Pi_i \overset{(1)}{=} \sum_{i=j}^{T} \frac{p_i v^{j-i}}{ik} - \sum_{i=j+1}^{T} \frac{p_i v^{j+1-i}}{ik} \\ &\overset{(3)}{=} \sum_{i=j}^{T} \frac{ip_1 v^{j-i}}{ik} - \sum_{i=j+1}^{T} \frac{ip_1 v^{j+1-i}}{ik} = \frac{p_1}{k}\left(\sum_{i=j}^{T} v^{j-i} - \sum_{i=j+1}^{T} v^{j+1-i}\right) \\ &= \frac{p_1}{k} v^{j-T}.\end{aligned}$$

By means of the analysis above, the objective function of (D) becomes

$$\begin{aligned}\sum_{j=1}^{T} jk\Pi_j &= \sum_{j=1}^{T} jk\frac{p_1}{k}v^{j-T} = \frac{p_1}{v^T}\sum_{j=1}^{T} jv^j = \frac{p_1}{v^T}\frac{v\left(Tv^{T+1}-(T+1)v^T+1\right)}{(v-1)^2}\\ &= p_1\frac{v^{T+2}\left(T-(T+1)v^{-1}+v^{-(T+1)}\right)}{v^{T+2}\left(1-2v^{-1}+v^{-2}\right)}\\ &= p_1\frac{T-(T+1)v^{-1}+v^{1-T}}{1-2v^{-1}+v^{-2}}.\end{aligned}$$

For $v \to \infty$, we have

$$\sum_{j=1}^{T} jk\Pi_j \to p_1 T \overset{(2)}{=} \frac{2}{T+1}.$$

This completes the proof. □

In the following section, we present deterministic and randomized online algorithms for OKIC and analyze their competitive ratios.

4 Competitive algorithms

In this section, deterministic and randomized online algorithms for the problem OKIC are discussed. The first, obvious choice for an online algorithm for OKIC is a greedy algorithm. In the following subsection, we show that a natural greedy algorithm is T-competitive and therefore best possible for $k = 1$.

4.1 A greedy algorithm

In each time period, the greedy algorithm for OKIC accepts as many requests as possible in a greedy manner with respect to the value of the requests.

Algorithm 1: Greedy Algorithm for OKIC.

```
1 for $t = 1, \ldots, T$ do
2     Accept the requests $r_i$ with $d_i = t$ in order of nonincreasing value until either no more requests
      are available or the capacity is fully utilized.
```

Theorem 3 *Algorithm 1 is T-competitive for the problem* OKIC *with arbitrary $k \in \mathbb{N}^+$.*

Proof Algorithm 1 always accepts the k requests of highest value denoted by $v_{(1)} \geq \cdots \geq v_{(k)}$ among all requests as there are at least k units of capacity available at each time period. OPT can accept no more than Tk requests in total. Since

$$\text{OPT} \leq \sum_{i=1}^{k} v_{(i)} + (Tk - k)v_{(k)} \leq \sum_{i=1}^{k} v_{(i)} + (T-1)\sum_{i=1}^{k} v_{(i)} = T\sum_{i=1}^{k} v_{(i)},$$

and $\text{ALG} \geq \sum_{i=1}^{k} v_{(i)}$, we have

$$\max_{\sigma} \frac{\text{OPT}(\sigma)}{\text{ALG}(\sigma)} \leq \frac{T\sum_{i=1}^{k} v_{(i)}}{\sum_{i=1}^{k} v_{(i)}} = T.$$

□

For $k = 1$, the competitive ratio of Algorithm 1 matches the lower bound of T for any deterministic algorithm as given in Theorem 1. For $k \geq 2$, the online player has more reach of play in order to outsmart the adversary. Thus, we are able to develop a better deterministic online algorithm for the case $k \geq 2$, which is presented in the next subsection.

4.2 A balancing algorithm

For $k \geq 2$, a deterministic online algorithm for OKIC with competitive ratio smaller than T can be constructed. The idea of the algorithm is as follows: in each time period, we set an upper bound on the number of requests that may be accepted. This upper bound increases over time in order to maximize the competitive ratio. In the first half of the time horizon, less than k requests may be accepted. This way, the online player is able to save up some capacity in order to hedge against the advantage of the offline player, which increases over time. In the second half of the time horizon, the saved up capacity is utilized and more than k requests may be accepted.

Algorithm 2: Balancing Algorithm for OKIC.

```
1 for $t = 1, \dots, T$ do
2     Set $R_t = \left\lceil \frac{t(2k-1)}{T+1} \right\rceil$;
3     Accept at most $R_t$ requests $r_i$ with $d_i = t$ in order of nonincreasing value until no more requests
      are available.
```

Theorem 4 *For $k \geq 2$, Algorithm 2 is c-competitive with $c = {}^{(T+1)k}/_{(2k-1)}$.*

Proof The proof is partitioned in two steps. First of all, we show that Algorithm 2 outputs a feasible solution, i.e., it is feasible to accept up to R_t requests in each time period t. Secondly, we prove that the algorithm achieves the claimed competitiveness.

To begin with, the feasibility of Algorithm 2 is established, i.e., in time period t at least R_t units of capacity are available. For this purpose, each time period $t \leq {}^{T}/_{2}$ is paired with the time period $T - t + 1$ and it is shown that in both time periods together not more then $2k$ requests and in time period t not more than k requests are accepted, which proves the feasibility.

For $t \leq {}^{T}\!/2$, we have

$$\frac{t(2k-1)}{T+1} + \frac{(T-t+1)(2k-1)}{T+1} = \frac{(T+1)(2k-1)}{T+1} = 2k-1, \tag{4}$$

i.e., the sum of R_t and R_{T-t+1} without ceiling functions is integral. We define

$$x_1 = R_t - \frac{t(2k-1)}{T+1} \quad \text{and} \quad x_2 = R_{T-t+1} - \frac{(T-t+1)(2k-1)}{T+1},$$

and, due to (4) and $k \in \mathbb{N}^+$, $x_1 + x_2 = 1$ or $x_1 + x_2 = 0$. Then, R_t and R_{T-t+1} add up to

$$\begin{aligned} R_t + R_{T-t+1} &= \left\lceil \frac{t(2k-1)}{T+1} \right\rceil + \left\lceil \frac{(T-t+1)(2k-1)}{T+1} \right\rceil \\ &= \frac{t(2k-1)}{T+1} + x_1 + \frac{(T-t+1)(2k-1)}{T+1} + x_2. \end{aligned}$$

Thus, we have

$$R_t + R_{T-t+1} = 2k - 1 + x_1 + x_2 \leq 2k.$$

Consequently, when considering the sum of the number of accepted requests in two time periods t and $T-t+1$ for $t \leq {}^{T}\!/2$, at most $2k$ requests are accepted. Additionally, for $t \leq {}^{T}\!/2$,

$$\left\lceil \frac{t(2k-1)}{T+1} \right\rceil \leq \left\lceil \frac{{}^{T}\!/2(2k-1)}{T+1} \right\rceil = \left\lceil \frac{T\,(k-{}^{1}\!/2)}{T+1} \right\rceil \leq \left\lceil k - \frac{1}{2} \right\rceil = k,$$

i.e., in each time period $t \leq {}^{T}\!/2$, at most k requests are accepted. Since in each time period k additional units of capacity are available, it is feasible to accept at most R_t requests in each time period t.

Now, the competitive ratio of Algorithm 2 is analyzed. Denote by OPT_t the total value of the items accepted by OPT in time period t and by ALG_t the total value of items accepted by Algorithm 2 in time period t. Since the number of items chosen by OPT in time period t is at most $t \cdot k$, Algorithm 2 recovers at least $\text{OPT}_t \cdot {}^{R_t}\!/_{tk}$. Thus, we have

$$\frac{\text{OPT}_t}{\text{ALG}_t} \leq \frac{\text{OPT}_t}{\text{OPT}_t \cdot {}^{R_t}\!/_{tk}} = \frac{tk}{R_t} = \frac{tk}{\left\lceil \frac{t(2k-1)}{T+1} \right\rceil} \leq \frac{(T+1)k}{2k-1}.$$

Since this holds for an arbitrary $t \in \{1, \ldots, T\}$, the competitive ratio of Algorithm 2 is given by ${}^{(T+1)k}\!/_{(2k-1)}$. □

Note that for $k \to \infty$ the competitive ratio of Algorithm 2 matches the lower bound on the competitive ratio of any randomized (and deterministic) algorithm, i.e.,

$$\lim_{k\to\infty} \frac{(T+1)k}{2k-1} = \frac{T+1}{2}.$$

4.3 A randomized greedy algorithm

In this subsection, a randomized online algorithm for the problem OKIC is presented. The idea behind Algorithm 3 is to act greedily with a certain probability in each time period. This way, the algorithm is eventually able to save up some capacity and at the same time cannot be leveraged by the adversary.

The probability of being greedy is adjusted in each time period in order to maximize the competitive ratio. In fact, the competitive ratio of Algorithm 3 matches the lower bound for any randomized online algorithm for OKIC given in Sect. 3.2. Basically, the probability of being greedy in a time period increases over time since we have to hedge against the capacity possibly saved up by the adversary, which also increases over time.

Algorithm 3: Randomized Greedy Algorithm for OKIC.

```
1 for $t = 1, \dots, T$ do
2     With probability $p_t = 2/(T-t+2)$, accept all requests $r_i$ with $d_i = t$ in order of nonincreasing
      value until either no more requests are available or the capacity is fully utilized. With probability
      $1 - p_t$, accept no requests at all.
```

Theorem 5 *Algorithm 3 is $(T+1)/2$-competitive for* OKIC *with $k \in \mathbb{N}^+$.*

Proof Denote by OPT_t the total value of the items accepted by OPT in time period t and by α_t the number of items accepted by Algorithm 3 in time period t. Since the number of items chosen by OPT in time period t is at most $t \cdot k$, Algorithm 3 recovers at least $\text{OPT}_t \cdot \alpha_t/tk$. Thus, the competitive ratio c is given by

$$c = \max_{\sigma} \frac{\text{OPT}(\sigma)}{\mathbb{E}\left[\text{ALG}(\sigma)\right]} \leq \max_{t=1,\dots,T} \frac{\text{OPT}_t}{\mathbb{E}\left[\text{OPT}_t \cdot \alpha_t/tk\right]} = \max_{t=1,\dots,T} \frac{tk}{\mathbb{E}\left[\alpha_t\right]},$$

where $\mathbb{E}[\alpha_t]$ denotes the expected number of accepted requests by ALG in time period t. We proceed by proving that

$$\mathbb{E}[\alpha_t] = p_t \left(\sum_{i=1}^{t-1} ikp_{t-i} \prod_{j=1}^{i-1} (1 - p_{t-j}) + tk \prod_{j=1}^{t-1} (1 - p_j) \right) \tag{5}$$

$$= \frac{2tk}{T+1}. \tag{6}$$

Equality (5) results from the following observation: in order to accept ik requests in time period t, first of all ALG has to accept requests in time period t, which happens with probability p_t. Additionally, ik units of capacity have to be available in time period t. Consequently, ALG has to reject accepting any requests in the previous $i-1$ time periods, which happens with probability $\prod_{j=1}^{i-1}\left(1-p_{t-j}\right)$, and accept the requests in time period $t-i$, which happens with probability p_{t-i}.

Furthermore, in order to accept tk requests in time period t, ALG has to reject accepting any requests in all previous periods, which happens with probability $\prod_{j=1}^{t-1}\left(1-p_j\right)$. Altogether, we end up with equality (5).

As a preliminary result for the proof of equality (6), we show by induction (see Induction I in the appendix) that

$$\prod_{j=1}^{t-1}\left(1-p_j\right)=\frac{(T-t+1)(T-t+2)}{T(T+1)},\quad \text{for } t=2,\ldots,T. \tag{7}$$

In a similar manner (see Induction II in the appendix), we show that

$$\prod_{j=1}^{i-1}\left(1-p_{t-j}\right)=\frac{(T-t+2)(T-t+1)}{(T-t+i)(T-t+i+1)},\quad \text{for } i=2,\ldots,T-1. \tag{8}$$

Then, by means of Eqs. (5), (7), and (8), we have

$$\begin{aligned}\mathbb{E}\left[\alpha_t\right]&=p_t\left(\sum_{i=1}^{t-1} ikp_{t-i}\prod_{j=1}^{i-1}\left(1-p_{t-j}\right)+tk\prod_{j=1}^{t-1}\left(1-p_j\right)\right)\\&=kp_t\left(\sum_{i=1}^{t-1}\frac{ip_{t-i}(T-t+2)(T-t+1)}{(T-t+i)(T-t+i+1)}+\frac{t(T-t+1)(T-t+2)}{T(T+1)}\right).\end{aligned} \tag{9}$$

Once again, we use induction (see Induction III in the appendix) to show that

$$\begin{aligned}\sum_{i=1}^{t-1}\frac{ip_{t-i}}{(T-t+i)(T-t+i+1)}&=\sum_{i=1}^{t-1}\frac{2i}{(T-t+i)(T-t+i+1)(T-t+i+2)}\\&=\frac{t(1-t)}{T(T+1)(t-T-1)},\quad \text{for } t=2,\ldots,T.\end{aligned} \tag{10}$$

With the help of Eqs. (9) and (10), the expected number of requests accepted by ALG in time period t with respect to sequence σ is now given by

$$\mathbb{E}\left[\alpha_t\right] = kp_t\left(\frac{t(1-t)(T-t+2)(T-t+1)}{T(T+1)(t-T-1)} + \frac{t(T-t+1)(T-t+2)}{T(T+1)}\right)$$
$$= -\frac{2tk(1-t)}{T(T+1)} + \frac{2tk(T-t+1)}{T(T+1)} = \frac{2tk}{T+1}.$$

Finally, by means of this result, the competitive ratio c becomes

$$c = \max_{t=1,\dots,T} \frac{tk}{\mathbb{E}\left[\alpha_t\right]} = \frac{tk}{\frac{2tk}{T+1}} = \frac{T+1}{2},$$

which completes the proof. □

5 Limited weights

In this section, we drop the assumption of *unit* weights and discuss the setting of k-incremental capacity with *limited* weights, i.e., we have weights $w_i \in \{1,\dots,k\}$. Let S_t denote the set of indices of requests accepted by some algorithm at time t. Then, the knapsack capacity c_t in time period t is now given as $c_t = c_{t-1} + k - \sum_{i \in S_t} w_i$ and $c_1 = k$.

Note that it is reasonable to consider *limited* weights $w_i \in \{1,\dots,k\}$ instead of *unlimited* weights $w_i \in \mathbb{N}^+$ since it it easy to see that the setting with unlimited weights does not admit any deterministic competitive online algorithms: If unlimited weights are allowed, presenting a request $r_1 = (1, 1, k)$ of value 1 and weight k in the first time period and then presenting a request $r_2 = (2, M, k+1)$ of very large value $M \in \mathbb{N}^+$ and weight $k+1$ in the second time period only if the online algorithm accepted r_1 shows that no deterministic online algorithm can be competitive.

5.1 Deterministic algorithms

Now, we consider a lower bound for any deterministic algorithm for OKIC with limited weights $w_i \in \{1,\dots,k\}$. For $k = 1$, we have unit weights and the results from Sects. 3 and 4 with $k = 1$ apply. For $k \geq 2$, we show the following lower bound on the competitive ratio of deterministic online algorithms:

Theorem 6 *For $k \geq 2$, no deterministic online algorithm for* OKIC *with limited weights can achieve a competitive ratio smaller than* $\left\lfloor Tk/\left(\left\lfloor \frac{k}{2} \right\rfloor + 1\right) \right\rfloor$.

Proof The proof is analogous to the proof of Theorem 1. For each time period $t = 1,\dots,T-1$, the adversary presents t identical requests $r_{t_1},\dots,r_{t_t}$ with

$$r_{t_i} = (t, v^t, k), \quad v \geq 1, \quad i = 1,\dots,t,$$

and, by the same argumentation as in the proof of Theorem 1, forces the online player to accept one request in each time period in order to be competitive.

In time period T, the adversary presents $\left\lfloor Tk/\left(\left\lfloor \frac{k}{2}\right\rfloor+1\right)\right\rfloor$ identical requests with value v^T and weight $\left\lfloor \frac{k}{2}\right\rfloor + 1$. Since the adversary did not accept any request before, he is able to accept all of these requests, whereas the online player is able to accept exactly one of these requests. This leads to a competitive ratio of

$$\frac{\text{OPT}}{\text{ALG}} = \frac{\left\lfloor \frac{Tk}{\left\lfloor \frac{k}{2}\right\rfloor+1}\right\rfloor v^T}{\sum_{j=1}^{T-1} v^j + v^T} \to \left\lfloor \frac{Tk}{\left\lfloor \frac{k}{2}\right\rfloor + 1}\right\rfloor \quad \text{for } v \to \infty.$$

□

The competitive ratio of the following algorithm matches the lower bound given in Theorem 6 for $k \to \infty$:

Algorithm 4: Greedy Algorithm for OKIC with limited weights.

```
1 for $t = 1, \dots, T$ do
2     Solve the knapsack problem with all requests $r_i$ with $d_i = t$ and the available capacity and
      accept the corresponding requests.
```

Note that the running time of Algorithm 4 is only pseudo-polynomial in the encoding length of the problem input as the algorithm has to solve a knapsack problem exactly in each time period. This can be done, e.g., by standard dynamic programming approaches (cf., for example, Martello and Toth 1990; Kellerer et al. 2004).

Theorem 7 *Algorithm 4 is $2T - 1$-competitive for* OKIC *with limited weights.*

Proof In each time period $t = 1, \dots, T$, OPT has at most tk units of capacity available. Hence, the requests accepted by OPT in time period t can be fractionally assigned to t knapsacks of size k each, such that at most $t-1$ requests overlap from one knapsack to the next, i.e., at most $t-1$ requests are fractionally assigned (cf. Fig. 1). Consequently, removing each of these requests and assigning it to its own additional knapsack yields an integral assignment of the requests accepted by OPT in time period t to at most $2t-1$ knapsacks. Since ALG has at least k units of capacity available in time period t, ALG obtains at least the value of the most valuable of these $2t-1$ knapsacks in period t. Since $2t-1 \leq 2T-1$ for every $t = 1, \dots, T$, this proves the desired competitive ratio. □

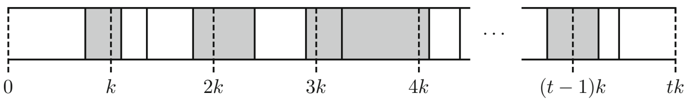

Fig. 1 Fractional assigment to t knapsacks of size k each. Fractionally assigned requests are shown in *grey*

For $k \to \infty$, the lower bound converges to the competitive ratio of Algorithm 4:

$$\left\lfloor \frac{Tk}{\lfloor \frac{k}{2} \rfloor + 1} \right\rfloor = \begin{cases} \left\lfloor \frac{2T}{1+\frac{2}{k}} \right\rfloor \to 2T-1 & \text{for } k \to \infty, \quad k \text{ even}, \\ \left\lfloor \frac{2T}{1+\frac{1}{k}} \right\rfloor \to 2T-1 & \text{for } k \to \infty, \quad k \text{ odd}. \end{cases}$$

5.2 Randomized algorithms

Finally, we consider randomized algorithms for OKIC with limited weights. By combining several methods used in the previous sections, a lower bound on the competitive ratio of any randomized online algorithm and a competitive randomized online algorithm can be established.

Theorem 8 *For $T \geq 2$ and $k \in \mathbb{N}^+$, no randomized algorithm for* OKIC *with limited weights $w_i \in \{1, \dots, k\}$ can achieve a competitive ratio smaller than $(T+1)/2$.*

Proof Consider the proof of Theorem 2. The request sequences σ_i consist of jk requests of the form $r_j = (j, v^j, 1)$ for each $j \in \{1, \dots, i\}$. We now replace these request sequences σ_i by request sequences consisting of j requests of the form $r_j = (j, v^j, k)$, for each $j \in \{1, \dots, i\}$. The remaining proof is then equivalent to the case of $k = 1$ in the proof for Theorem 2. □

In order to construct a competitive online algorithm for OKIC with limited weights, we combine the methods used in Algorithms 3 and 4.

Algorithm 5: Randomized Greedy Algorithm for OKIC with limited weights.

```
1 for $t = 1, \dots, T$ do
2     With probability $p_t = 2/(T-t+2)$, solve the knapsack problem with all requests $r_i$ with $d_i = t$
      and the available capacity and accept the corresponding requests. With probability $1 - p_t$, accept
      no requests at all.
```

Theorem 9 *Algorithm 5 is* $3\,((T+1)/2)$*-competitive for* OKIC *with limited weights.*

Proof First of all, note that, in each time period, it is advantageous for the adversary to reveal additional invaluable requests with appropriate weights such that the online player uses up all available capacity when solving the knapsack problem. Thus, the expected available capacity $\mathbb{E}_c^t$ in time period t is given by

$$\mathbb{E}_c^t = \sum_{i=1}^{t-1} ik \cdot p_{t-i} \prod_{j=1}^{i-1} \left(1 - p_{t-j}\right) + tk \prod_{j=1}^{t-1} \left(1 - p_j\right).$$

See the proof of Theorem 5 for a detailed explanation. By Eq. (6) and the definition of p_t, we have

$$\mathbb{E}_c^t = \frac{2tk}{(T+1)\,p_t} = \frac{tk\,(T-t+2)}{T+1}. \tag{11}$$

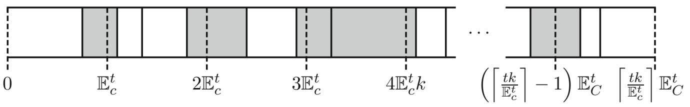
...
0
$\mathbb{E}_c^t$
$2\mathbb{E}_c^t$
$3\mathbb{E}_c^t$
$4\mathbb{E}_c^t k$
$\left(\left\lceil \frac{tk}{\mathbb{E}_c^t}\right\rceil - 1\right)\mathbb{E}_C^t$
$\left\lceil \frac{tk}{\mathbb{E}_c^t}\right\rceil \mathbb{E}_C^t$

Fig. 2 Fractional assigment to $\lceil tk/\mathbb{E}_c^t\rceil$ knapsacks of size $\mathbb{E}_c^t$ each. Fractionally assigned requests are shown in *grey*

Now we apply the same line of argument as in the proof of Theorem 7, but incorporate the expected available capacity. In each time period $t = 1, \ldots, T$, OPT has at most tk units of capacity available. Hence, the requests accepted by OPT in time period t can be fractionally assigned to $\lceil tk/\mathbb{E}_c^t\rceil$ knapsacks of size $\mathbb{E}_c^t \geq k$ each, such that at most $\lceil tk/\mathbb{E}_c^t\rceil - 1$ requests overlap from one knapsack to the next, i.e., at most $\lceil tk/\mathbb{E}_c^t\rceil - 1$ requests are fractionally assigned (cf. Fig. 2). Consequently, removing each of these requests and assigning it to its own additional knapsack yields an integral assignment of the requests accepted by OPT in time period t to at most $2\lceil tk/\mathbb{E}_c^t\rceil - 1$ knapsacks. Since the expected available capacity of ALG in time period t is given by $\lceil tk/\mathbb{E}_c^t\rceil$, ALG obtains at least the value of the most valuable of these $2\lceil tk/\mathbb{E}_c^t\rceil - 1$ knapsacks in period t. Hence, we obtain

$$\begin{aligned}\frac{\text{OPT}}{\mathbb{E}\left[\text{ALG}\right]} &\leq \frac{\sum_{t=1}^{T}\sum_{i\in\mathcal{A}_t^{\text{OPT}}} v_i}{\sum_{t=1}^{T}\frac{p_t}{2\lceil tk/\mathbb{E}_c^t\rceil-1}\sum_{i\in\mathcal{A}_t^{\text{OPT}}} v_i} \\ &\leq \frac{\sum_{t=1}^{T}\sum_{i\in\mathcal{A}_t^{\text{OPT}}} v_i}{\frac{2}{3T+3}\sum_{t=1}^{T}\sum_{i\in\mathcal{A}_t^{\text{OPT}}} v_i} = 3\left(\frac{T+1}{2}\right).\end{aligned}$$

Here, the second inequality holds since, for all $1 \leq t \leq T$, we have:

$$\frac{p_t}{2\left\lceil \frac{tk}{\mathbb{E}_c^t}\right\rceil - 1} \overset{(11)}{=} \frac{\frac{2}{T-t+2}}{2\left\lceil \frac{T+1}{T-t+2}\right\rceil - 1} \geq \frac{\frac{2}{T-t+2}}{2\left(\frac{T+1}{T-t+2}+1\right) - 1} = \frac{2}{3T+4-t} \geq \frac{2}{3T+3}.$$

□

6 Removable items

The lower bounds presented in Sect. 3 and the competitive ratios of the algorithms presented in Sects. 4 and 5 are all dependent on the time horizon T. Thus, for $T \to \infty$, all those algorithms are not competitive, and, even worse, there are no competitive algorithms at all due to the dependence of the lower bounds on T.

Therefore, we now increase the power of the online player by allowing the online player to remove previously accepted items from the knapsack in any time period. Once an item is removed from the knapsack it cannot be accepted again. We consider the case of k-incremental capacity and limited weights, i.e., $w_i \in \{1, \ldots, k\}$. The problem is in the following referred to as OKIC *with removable items*.

Algorithm 6 given below is based on the linear relaxation of the offline version of OKIC with removable items.

Algorithm 6: Algorithm for OKIC with removable items.

```
1  for $t = 1, \dots, T$ do
2      Let $\mathcal{N}^t$ be the set of new items (possibly fractionally) accepted by IFK$^t$ in time period $t$.
3      if $\alpha_t^t = 1$ then
4          Accept all items in $\mathcal{N}^t$.
5          Remove items accepted in previous time periods in order of nondecreasing efficiency such
           that the capacity constraints are satisfied.
6      else
7          if $\sum_{i=1}^{s_t^t - 1} (v_t)_i \geq (v_t)_{s_t^t}$ then
8              Accept items $i = 1, \dots, s_t^t - 1$.
9              Remove items accepted in previous time periods in order of nondecreasing efficiency
               such that the capacity constraints are satisfied.
10         else
11             Accept the split item $s_t^t$.
```

For $1 \leq t \leq T$, we define the *incremental fractional knapsack problem with time horizon* t, denoted by IFK^t. For each time period $\tau \in \{1, \dots, t\}$ and each item $i \in \{1, \dots, n_\tau\}$, we introduce continuous variables $0 \leq \left(x_\tau^t\right)_i \leq 1$ that take value 1 if the corresponding item is accepted and 0 otherwise. Here, n_τ denotes the number of items in time period τ, $(v_\tau)_i$ denotes the value of the i-th item given in time period τ, and $(w_\tau)_i$ the corresponding weight. The problem IFK^t is then given by

$$\begin{aligned} \max \quad & \sum_{\tau=1}^{t} \sum_{i=1}^{n_\tau} \left(x_\tau^t\right)_i (v_\tau)_i && \text{(IFK}^t\text{)} \\ \text{s.t.} \quad & \sum_{\tau=1}^{\bar{t}} \sum_{i=1}^{n_\tau} \left(x_\tau^t\right)_i (w_\tau)_i \leq k \cdot \bar{t} \qquad \text{for } \bar{t} = 1, \dots, t, \\ & 0 \leq \left(x_\tau^t\right)_i \leq 1 \qquad \text{for } i = 1, \dots, n_\tau,\ \tau = 1, \dots, t. \end{aligned}$$

Note that IFK^t denotes the incremental fraction knapsack problem with t time periods, i.e., $\tau = 1, \dots, t$.

The optimal solution of IFK^t is given by accepting the most efficient items while respecting the capacity constraint of each time period. In the following, assume that, in each time period τ, the new items are sorted by efficiency, i.e., $(v_\tau)_1/(w_\tau)_1 \geq (v_\tau)_2/(w_\tau)_2 \geq \cdots \geq (v_\tau)_{n_\tau}/(w_\tau)_{n_\tau}$. Then, the optimal solution vector x^t of IFK^t is given by $x^t = \left(x_1^t, \dots, x_t^t\right)$, where each $x_\tau^t \in \mathbb{R}_+^{n_\tau}$ for $1 \leq \tau \leq t$ is given by

$$x_\tau^t = \left(\underbrace{1, \dots, 1}_{s_\tau^t - 1}, \alpha_\tau^t, 0, \dots, 0 \right), \quad \text{with } 0 < \alpha_\tau^t \leq 1, \tag{12}$$

where for some $1 \leq s_\tau^t \leq n_\tau$ (note that $s_t^t = |\mathcal{N}^t|$, see Algorithm 6). Note that it is possible that x_τ^t does not use the full $k \cdot \tau$ units of capacity available in time period τ since it may be beneficial to safe some capacity for items of higher efficiency arriving in later periods. The item (possibly fractionally) accepted to an amount α_τ^t in time period τ will be referred to as the *split item* of time period τ. Observe that $s_\tau^t \leq s_\tau^{t'}$ for all $t \geq t'$, since IFK^t, for $t > t'$, will never accept any item of time period τ that was not accepted by $\text{IFK}^{t'}$.

We now analyze the competitiveness of Algorithm 6, also referred to as ALG in the proof of the following theorem.

Theorem 10 *For $k \geq 2$, Algorithm 6 is 3-competitive for* OKIC *with removable items. For $k = 1$, Algorithm 6 finds the optimal offline solution.*

Proof First of all, we consider the case $k = 1$. In this case, IFK^T never accepts any item fractionally, i.e., $\alpha_\tau^T = 1$ for all $\tau = 1, \ldots, T$. It is easy to see that the solution produced by ALG is identical to the solution of IFK^T. Since $\text{IFK}^T \geq \text{OPT}$, ALG obtains an optimal solution for $k = 1$.

For $k \geq 2$, consider now the incremental fractional knapsack problem with time horizon T, i.e., IFK^T: Since $\text{IFK}^T \geq \text{OPT}$, it suffices to prove that $3 \cdot \text{ALG} \geq \text{IFK}^T$ for $k \geq 2$.

For $1 \leq t \leq T$, denote by ALG^t the online algorithm after time period t and by $y^t = \left(y_1^t, \ldots, y_t^t\right)$ the solution vector produced by ALG^t, where $y_\tau^t \in \mathbb{R}_+^{n_\tau}$ for $1 \leq \tau \leq t$. In the following, we prove by induction on t that, for each time period t and the corresponding program IFK^t, we have $3 \cdot \text{ALG}^t \geq \text{IFK}^t$. For $t = T$, we then have $3 \cdot \text{ALG} \geq \text{IFK}^T \geq \text{OPT}$. First of all, we show that this holds for $t = 1$ and $t = 2$. The step from an arbitrary $t - 1$ to t then works analogously.

Without loss of generality, assume that the weight of the new items in each time period is larger than k: If the weight of new items in time period t is at most k, $\mathcal{N}^t$ contains all new items of time period t and $\alpha_t^t = 1$. Hence, by Step 6 of Algorithm 6, ALG accepts all new items in addition to the items in the knapsack after the previous time period and the adversary is not able to gain a competitive edge over the online player (note that ALG does not remove any items in Step 6).

Consider $t = 1$ and the optimal solution vector $x^1 = \left(x_1^1\right)$ for IFK^1, where x_1^1 is given by

$$x_1^1 = \left(\underbrace{1, \ldots, 1}_{s_1^1 - 1}, \alpha_1^1, 0, \ldots, 0 \right), \tag{13}$$

whereas the solution vector y^1 of ALG_1 is given by $y^1 = \left(y_1^1\right)$, where

$$y_1^1 = \left(\underbrace{1, \ldots, 1}_{s_1^1 - 1}, 0, 0, \ldots, 0 \right) \text{ if } \sum_{i=1}^{s_1^1 - 1} (v_1)_i \geq (v_1)_{s_1^1}, \tag{14}$$

or

$$y_1^1 = \left(\underbrace{0, \ldots, 0}_{s_1^1 - 1}, 1, 0, \ldots, 0 \right) \text{ if } \sum_{i=1}^{s_1^1 - 1} (v_1)_i < (v_1)_{s_1^1} . \tag{15}$$

Since ALG^1 accepts the more valuable solution, we have

$$2 \cdot \text{ALG}^1 \geq \text{IFK}^1. \tag{16}$$

Figures 3 and 4 depict the possible situations after the first time period.

Note that we can neglect items that are rejected by both IFK^1 and ALG^1 since the online algorithm cannot use them in later time periods and $s_1^t \leq s_1^1$ for $t \geq 1$.

Now, consider $t = 2$. We distinguish two cases with respect to α_2^2: Either the split item of the second time period is completely accepted, i.e., $\alpha_2^2 = 1$, or the split item of the second time period is fractionally accepted, i.e., $\alpha_2^2 < 1$.

Case 1: $\alpha_2^2 = 1$.

In this case, the split item of the second time period is completely accepted by IFK^2. Consequently, ALG^2 accepts all items in $\mathcal{N}_2$, according to Step 6 of Algorithm 6, and we have

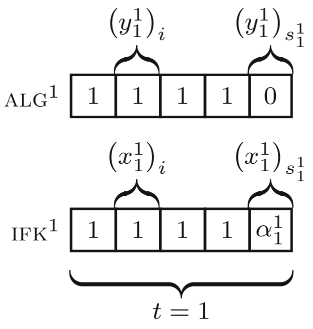

Fig. 3 ALG^1 according to (14)

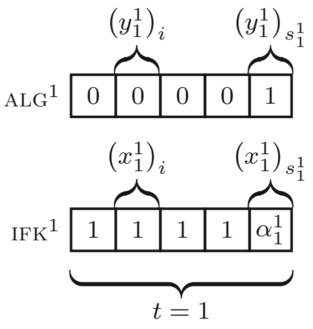

Fig. 4 ALG^1 according to (15)

$$\underbrace{\sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_2)_i}_{\substack{\text{value of new items}\\ \text{accepted by ALG}^2}} = \underbrace{\sum_{i=1}^{n_2} \left(x_2^2\right)_i (v_2)_i}_{\substack{\text{value of new items}\\ \text{accepted by IFK}^2}} . \tag{17}$$

Now, consider the items from the first time period. ALG^2 possibly has to remove some of the items accepted in the first time period in order to give way for the accepted items from the second time period (see Step 6 of Algorithm 6). We distinguish two cases with respect to the behavior of ALG^1:

Case 1.1: ALG^1 acted according to (14).

In this case, ALG^1 accepted all items that are accepted by IFK^1 in the first time period except for the split item s_1^1. If ALG^2 does not have to remove item s_1^1 from the first time period, we have $2 \cdot \text{ALG}^2 \geq \text{IFK}^2$ due to (16) and (17).

Therefore, assume that ALG^2 has to remove item s_1^1 from the first time period and, hence, the weight of items from the second time period accepted by ALG^2 is larger than k. The remaining value for ALG^2 of items from the first time period is then given by

$$\sum_{i=1}^{n_1} \left(y_1^2\right)_i (v_1)_i = \sum_{i=1}^{s_1^1-1} (v_1)_i - \sum_{i=d}^{s_1^1-1} (v_1)_i ,$$

for some $1 \leq d \leq s_1^1 - 1$ (note that the items in each time period are sorted by nonincreasing efficiency). However, since ALG^2 has to remove items $d, \dots, s_1^1 - 1$ from the first time period and both ALG^2 and IFK^2 accept the same items from the second time period, IFK^2 cannot accept more than items $1, \dots, d$ from the first time period due to capacity constraints, i.e., $s_1^2 = d$ (see Fig. 5). Therefore, we have

$$\sum_{i=1}^{n_1} \left(y_1^2\right)_i (v_1)_i = \sum_{i=1}^{d-1} (v_1)_i = \sum_{i=1}^{s_1^2-1} \left(x_1^2\right)_i (v_1)_i . \tag{18}$$

The weight of the items from the second time period accepted by ALG^2 is larger than k and, additionally, the efficiency of each item from the second time period accepted by ALG^2 is at least as large as the efficiency of item s_1^2 (otherwise, IFK^2 would not have accepted all new items completely). Since $s_1^2 \leq s_1^1$, this holds also for s_1^1. Thus, we have

$$\sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_1)_i \geq (v_1)_{s_1^1} \quad \text{and} \quad \sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_1)_i \geq (v_1)_{s_1^2} . \tag{19}$$

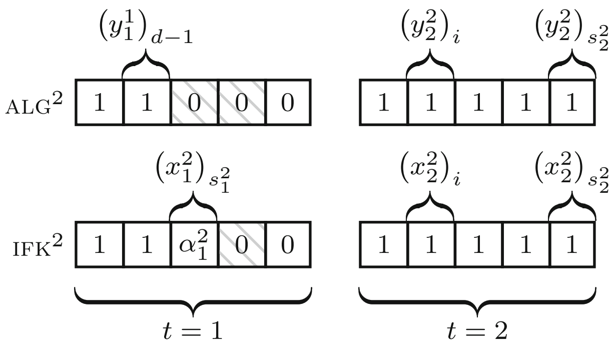

Fig. 5 Solution vectors of ALG^2 and IFK^2 in Case 1.1. Hatched fields denote removed items

For the total value of ALG^2, we then have

$$\begin{aligned}
2\cdot \text{ALG}^2 &= 2\left(\sum_{i=1}^{n_1}\left(y_1^2\right)_i (v_1)_i + \sum_{i=1}^{n_2}\left(y_2^2\right)_i (v_2)_i\right)\\
&\overset{(17)}{=} 2\sum_{i=1}^{n_1}\left(y_1^2\right)_i (v_1)_i + \sum_{i=1}^{n_2}\left(y_2^2\right)_i (v_2)_i + \sum_{i=1}^{n_2}\left(x_2^2\right)_i (v_2)_i\\
&\overset{(18)}{=} 2\sum_{i=1}^{s_1^2-1}\left(x_1^2\right)_i (v_1)_i + \sum_{i=1}^{n_2}\left(y_2^2\right)_i (v_2)_i + \sum_{i=1}^{n_2}\left(x_2^2\right)_i (v_2)_i\\
&\overset{(19)}{\geq} 2\sum_{i=1}^{s_1^2-1}\left(x_1^2\right)_i (v_1)_i + (v_1)_{s_1^2} + \sum_{i=1}^{n_2}\left(x_2^2\right)_i (v_2)_i\\
&\geq \sum_{i=1}^{n_1}\left(x_1^2\right)_i (v_1)_i + \sum_{i=1}^{n_2}\left(x_2^2\right)_i (v_2)_i\\
&= \text{IFK}^2.
\end{aligned}$$

Case 1.2: ALG^1 acted according to (15).

In this case, ALG^1 accepted only the split item s_1^1 in the first time period. If ALG^2 does not have to remove item s_1^1 from the first time period, we have $2\cdot \text{ALG}^2 \geq \text{IFK}^2$ due to (16) and (17).

Therefore, assume that ALG^2 has to remove item s_1^1 from the first time period and, hence, the weight of items from the second time period accepted by ALG^2 is larger than k. We then have

$$\sum_{i=1}^{n_1}\left(y_1^2\right)_i (v_1)_i = 0, \tag{20}$$

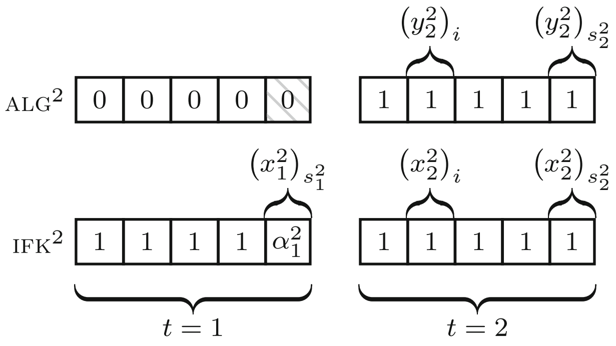

Fig. 6 Solution vectors of ALG^2 and IFK^2 in Case 1.2. Hatched fields denote removed items

see also Fig. 6. Furthermore, (19) holds by the same arguments as in Case 1.1. In this case, the total value of ALG^2 satisfies

$$\begin{aligned}
3 \cdot \text{ALG}^2 &= 3\left(\sum_{i=1}^{n_1} \left(y_1^2\right)_i (v_1)_i + \sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_2)_i\right) \\
&\overset{(20)}{=} 3\sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_2)_i \\
&\overset{(17)}{=} 2\sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_2)_i + \sum_{i=1}^{n_2} \left(x_2^2\right)_i (v_2)_i \\
&\overset{(19)}{\geq} (v_1)_{s_1^1} + (v_1)_{s_1^2} + \sum_{i=1}^{n_2} \left(x_2^2\right)_i (v_2)_i \\
&\overset{(15)}{\geq} \sum_{i=1}^{s_1^1-1} \left(x_1^1\right)_i (v_1)_i + (v_1)_{s_1^2} + \sum_{i=1}^{n_2} \left(x_2^2\right)_i (v_2)_i \\
&\geq \sum_{i=1}^{s_1^2} \left(x_1^2\right)_i (v_1)_i + \sum_{i=1}^{n_2} \left(x_2^2\right)_i (v_2)_i \quad (\text{since } s_1^2 \leq s_1^1) \\
&= \text{IFK}^2.
\end{aligned}$$

Case 2: $\alpha_2^2 < 1$.

In this case, the split item of the second time period is only fractionally accepted by IFK^2. Therefore, ALG^2 has to decide whether to take the split item s_2^2, or all new items accepted by IFK^2 except for the split item. Since ALG^2 accepts the more valuable solution, we have

$$\underbrace{2\sum_{i=1}^{n_2}\left(y_2^2\right)_i (v_2)_i}_{\substack{\text{value of new items}\\ \text{accepted by ALG}^2}} \geq \underbrace{\sum_{i=1}^{n_2}\left(x_2^2\right)_i (v_2)_i}_{\substack{\text{value of new items}\\ \text{accepted by IFK}^2}} . \tag{21}$$

If ALG^2 accepts only the split item s_2^2, there is no need for ALG^2 to remove any of the items accepted in the first time period since the weight of each item is at most k and there are k units of additional capacity available. Thus, we have $2 \cdot$ ALG$^2 \geq$ IFK^2 due to (16) and (21).

Therefore, assume that ALG^2 accepts all new items of the second time period accepted by IFK^2 except for the split item s_2^2. Before we proceed, we make the following observation:

Observation 1 Since IFK^2 accepted the split item s_2^2 fractionally, the efficiency of the split item is the lowest among all items accepted by IFK^2. In particular, IFK^2 accepts each item from the first time period accepted by IFK^1 to the same fraction (if its efficiency is higher than the efficiency of s_2^2), or IFK^2 does not accept the item at all (if its efficiency is lower than the efficiency of s_2^2). For items with equal efficiency, we can assume without loss of generality that IFK^2 is the offline solution that assigns the fractionality only to s_2^2, since this does not change the value of the solution of IFK^2.

As in the previous case, we now distinguish two cases with respect to the behavior of ALG^1:

Case 2.1: ALG^1 acted according to (14).

In this case, ALG^1 accepted all items that are accepted by IFK^1 in the first time period except for the split item s_1^1. If ALG^2 does not have to remove item s_1^1 from the first time period, we have $2 \cdot$ ALG$^2 \geq$ IFK^2 due to (16) and (21).

Therefore, assume that ALG^2 has to remove item s_1^1 from the first time period and, hence, the weight of items from the second time period accepted by ALG^2 is larger than k. As in Case 1.1, the remaining value for ALG^2 of items from the first time period is then given by

$$\sum_{i=1}^{n_1}\left(y_1^2\right)_i (v_1)_i = \sum_{i=1}^{s_1^1-1} (v_1)_i - \sum_{i=d}^{s_1^1-1} (v_1)_i ,$$

for some $1 \leq d \leq s_1^1 - 1$. Due to Observation 1, IFK^2 accepts at most items $1, \ldots, d-1$ in this case, i.e., $s_1^2 \leq d-1$, since IFK^2 uses at least as much capacity as ALG^2 for items from the second time period (see Fig. 7). Thus, we have

$$\sum_{i=1}^{n_1}\left(y_1^2\right)_i (v_1)_i = \sum_{i=1}^{s_1^2}\left(x_1^2\right)_i (v_1)_i , \tag{22}$$

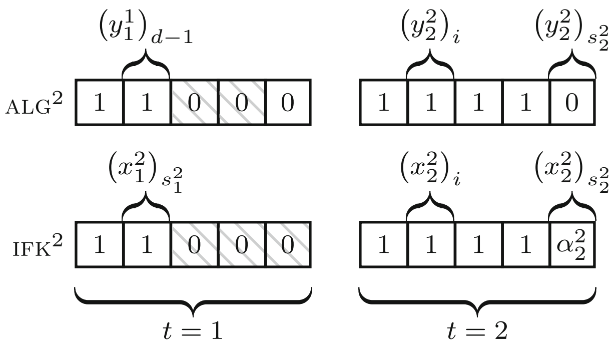

Fig. 7 Solution vectors of ALG^2 and IFK^2 in Case 2.1. Hatched fields denote removed items

and obtain $2 \cdot \text{ALG}^2 \geq \text{IFK}^2$ by means of (21) and (22).

Case 2.2: ALG^1 acted according to (15).

In this case, ALG^1 accepted only the split item s_1^1 in the first time period. If ALG^2 does not have to remove item s_1^1 from the first time period, we have $2 \cdot \text{ALG}^2 \geq \text{IFK}^2$ due to (16) and (21).

Therefore, assume that ALG^2 has to remove item s_1^1 from the first time period and, hence, the weight of items from the second time period accepted by ALG^2 is larger than k. As in Case 1.2, we have

$$\sum_{i=1}^{n_1} \left(y_1^2\right)_i (v_1)_i = 0. \tag{23}$$

The weight of new items accepted by IFK^2 must also be larger than k and, due to Observation 1, IFK^2 does not accept s_1^1. Therefore, we have

$$\sum_{i=1}^{n_1} \left(x_1^2\right)_i (v_1)_i \leq \sum_{i=1}^{s_1^1-1} (v_1)_i\,, \tag{24}$$

see also Fig. 8. Additionally, the efficiency of each item from the second time period accepted by ALG^2 is at least as large as the efficiency of item s_1^1 (otherwise, IFK^2 would not have accepted new items with weight larger than k). Thus, we have

$$\sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_1)_i \geq (v_1)_{s_1^1} \overset{(15)}{>} \sum_{i=1}^{s_1^1-1} (v_1)_i \overset{(24)}{\geq} \sum_{i=1}^{n_1} \left(x_1^2\right)_i (v_1)_i\,. \tag{25}$$

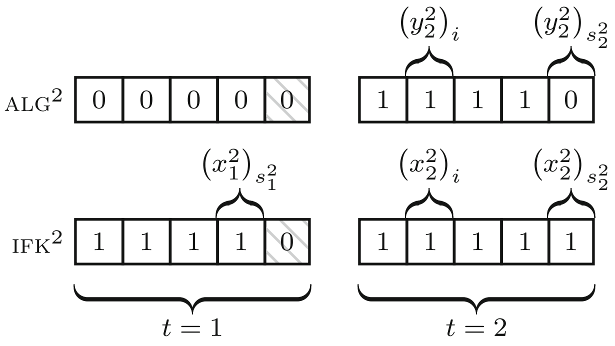

Fig. 8 Solution vectors of ALG^2 and IFK^2 in Case 2.2. Hatched fields denote removed items

For the total value of ALG^2, we then have

$$\begin{aligned}
3 \cdot \text{ALG}^2 &= 3\left(\sum_{i=1}^{n_1} \left(y_1^2\right)_i (v_1)_i + \sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_2)_i\right) \\
&\overset{(23)}{=} 3\sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_2)_i \\
&\overset{(21)}{\geq} \sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_2)_i + \sum_{i=1}^{n_2} \left(x_2^2\right)_i (v_2)_i \\
&\overset{(25)}{\geq} \sum_{i=1}^{n_1} \left(x_1^2\right)_i (v_1)_i + \sum_{i=1}^{n_2} \left(x_2^2\right)_i (v_2)_i \\
&= \text{IFK}^2.
\end{aligned}$$

Consider now the step from an arbitrary $t-1$ to t for $t > 2$. The basic analysis works analogously, but we have to make some comments concerning the details.

The analysis is identical with respect to the new items accepted in time period t by ALG^t and IFK^t. If no items accepted by ALG^{t-1} are removed by ALG^t, we are done. Thus, assume that ALG^t has to remove items accepted by ALG^{t-1}. In contrast to the step from the first to the second time period, ALG^t possibly removes items from several time periods. A generalization of Observation 1 helps us to analyze this situation:

Observation 2 If IFK^t accepts the split item of time period t fractionally, each item accepted by IFK^{t-1} is either accepted by IFK^t to the same fraction, or not accepted at all by IFK^t.

Observation 3 If IFK^t accepts the split item of time period t completely, there is at most one item accepted by IFK^{t-1} that is accepted by IFK^t to a smaller but positive fraction compared to IFK^{t-1}.

Both observations hold due to the same argumentation as for Observation 1.

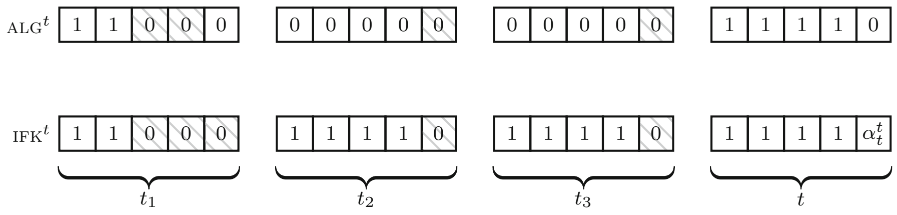

Fig. 9 Solution vectors of ALG^t and IFK^t. Hatched fields denote removed items

Now, consider the case that IFK^t accepts the split item of time period t fractionally. Due to Observation 2, each time period $t' < t$ for which ALG$^{t'}$ acted according to (14) can be analyzed as in Case 2.1 (see (22) and t_1 in Fig. 9). Each time period for which ALG$^{t'}$ acted according to (15) is analyzed as in Case 2.2 (see t_2 and t_3 in Figure 9), with the following additional argument: Before ALG^t removes the split item from time period t', at least the k additional units of capacity from time period t are used up by new items with efficiency at least as high as the split item's efficiency. After ALG^t removed the split item, the corresponding block of k units of capacity is available. Note that this must be at least k units of capacity, since $\alpha_{t'}^{t'} < 1$ in time period t'. Thus, before the next split item is removed by ALG^t, again k units of capacity are used by new items with efficiency at least as high as the previous items, and so forth. Therefore, the total value of all split items removed by ALG^t is not larger than the total value of all new items accepted by ALG^t.

Finally, consider the case that IFK^t accepts the split item of time period t completely. Due to Observation 3, there is at most one item accepted by IFK^{t-1} that is accepted by IFK^t to a smaller but positive fraction compared to IFK^{t-1}. If we neglect this item, the analysis works as in the case above. Since ALG^t accepts all new items that are accepted by IFK^t, ALG^t obtains the same value from new items as IFK^t, instead of only half of the value as in the case above. Since the neglected item has value not larger than total value of the new items of IFK^t, the value of the new items accepted by ALG^t is at least half of the value of both the new items accepted by IFK^t and the value of the neglected item. The value of the items from previous time periods except for the neglected item can be bounded as before. This concludes the proof. □

In the following, we prove that the analysis of Algorithm 6 is tight.

Theorem 11 *For $k \geq 3$, the competitive ratio of Algorithm 6 is 3.*

Proof Consider the requests $r_1 = (1, k, 1)$, $r_2 = (1, k + \epsilon, k)$, $\epsilon > 0$, and $r_3 = (1, k - 2, k - 2)$, that are presented to ALG in the first time period. Note that $k \geq 3$. We have an ordering with respect to efficiency of

$$\frac{v_1}{w_1} = \frac{k}{1} > \frac{(k+\epsilon)}{k} = \frac{v_2}{w_2} > 1 = \frac{v_3}{w_3}$$

for ϵ sufficiently small. Thus, IFK^1 accepts r_1 and a fraction of ${}^{k-1}/_{k}$ of r_2. Since $w_1 + w_2 = 1 + k > k$ and $v_2 = k + \epsilon > k = v_1$, ALG accepts r_2.

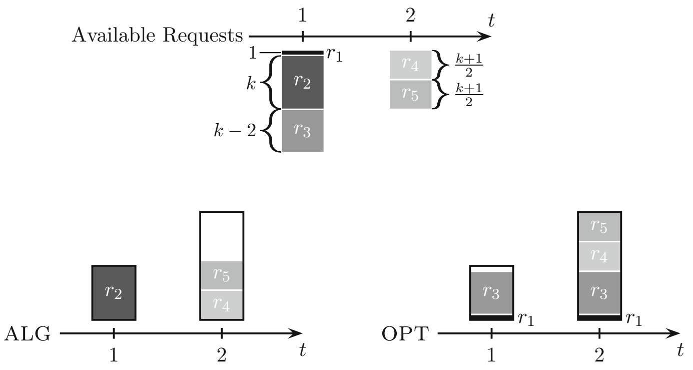

Fig. 10 Lower bound for OKIC with removable items for $k \geq 2$ odd

Assume that k is odd. Then, in the second time period, two identical requests

$$r_4 = r_5 = \left(2, \frac{(k+1)(k+2\epsilon)}{2k}, \frac{k+1}{2}\right)$$

are revealed (see also Fig. 10). The efficiency order is given by

$$\frac{v_1}{w_1} = \frac{k}{1} > \frac{v_4}{w_4} = \frac{v_5}{w_5} = \frac{\frac{(k+1)(k+2\epsilon)}{2k}}{\frac{k+1}{2}} = \frac{k+2\epsilon}{k} > \frac{k+\epsilon}{k} = \frac{v_2}{w_2}.$$

IFK^2 accepts r_1, r_4, r_5, and a fraction of $k-2/k$ of r_2. Thus, ALG accepts r_4 and r_5. Since $w_4 + w_5 + w_2 = 2k + 1 > 2k$, ALG has to remove r_2. $w_4 + w_5 = k + 1 < 2k$ and $w_4 + w_5 + w_2 = 2k + 1 > 2k$. Therefore, S_2 is now given by $S_2 = \{v_2, v_4, v_5\}$. The value of ALG is then given by

$$\text{ALG} = v_4 + v_5 = \frac{(k+1)(k+2\epsilon)}{k} = k + 1 + 2\epsilon + \frac{2\epsilon}{k}.$$

The optimal offline solution is given by accepting r_1,r_3,r_4, and r_5. This solution is feasible since $w_1 + w_3 = k - 1$ and $w_1 + w_3 + w_4 + w_5 = 2k$, and the total value is given by

$$\text{OPT} = v_1 + v_3 + v_4 + v_5 = k + k - 2 + \frac{(k+1)(k+2\epsilon)}{k} = 3k - 1 + 2\epsilon + \frac{2\epsilon}{k}.$$

Since $\epsilon > 0$ can be chosen small, the competitive ratio is in this case given by $\text{OPT}/\text{ALG} = (3k-1)/(k+1)$, and for $k \to \infty$, we have $\text{OPT}/\text{ALG} \to 3$.

It remains to prove the theorem for even k. For this case, we slightly change the weight and value of items r_4 and r_5, i.e., we set

$$r_4 = \left(2, \frac{k+2\epsilon}{2}, \frac{k}{2}\right) \quad \text{and} \quad r_5 = \left(2, \frac{k+2\epsilon}{2}\left(1 + \frac{2}{k}\right), \frac{k}{2} + 1\right).$$

Since

$$\frac{v_1}{w_1} = \frac{k}{1} > \frac{v_4}{w_4} = \frac{v_5}{w_5} = \frac{k+2\epsilon}{k} > \frac{k+\epsilon}{k} = \frac{v_2}{w_2},$$

ALG again accepts r_4 and r_5 in the second time period and, due to $w_4 + w_5 + w_2 = 2k + 1 > 2k$, ALG has to remove r_2. The value of ALG is then given by

$$v_4 + v_5 = \frac{k+2\epsilon}{2}\left(2 + \frac{2}{k}\right) = k + 1 + 2\epsilon + \frac{2\epsilon}{k}.$$

The optimal offline solution is given by accepting r_1,r_3,r_4, and r_5. This solution is still feasible and features the same total value as in the case of odd k. Therefore, the competitive ratio of Algorithm 6 is three. □

In the following, lower bounds on the competitive ratio of any deterministic online algorithm for OKIC with removable items are provided.

Theorem 12 *For $k \geq 4$, no deterministic online algorithm for* OKIC *with removable items can achieve a competitive ratio smaller than $\sqrt{2}$. For $k = 3$ and $k = 2$, lower bounds are given by $(1+\sqrt{10})/3$ and $(1+\sqrt{17})/4$, respectively.*

Proof Consider the request sequence $\sigma = (a_1, b_1, a_2, b_2, \ldots, a_k, b_k)$. In the first time period $t = 1$, the adversary reveals $a_1 = (1, 1, 1)$ and $b_1 = (1, x, k)$ with $x > 1$. Obviously, the online player must accept either a_1 or b_1 in order to be competitive. If the online player accepts a_1, all further requests a_t and b_t, for $t = 2, \ldots, k$, will be worthless and the competitive ratio is given by x since the adversary chooses b_1.

Otherwise, the online player accepts b_1 and the adversary reveals $a_2 = (1, 1, 1)$ and $b_2 = (1, x, k)$. Again, if the online player accepts a_2, all further requests a_t and b_t, for $t = 3, \ldots, k$, will be worthless and the competitive ratio is given by $(x+2)/(x+1)$ since the adversary accepts a_1 in the first period and a_2 and b_2 in the second period. Otherwise, if the online player accepts b_2, the adversary reveals $a_3 = (1, 1, 1)$ and $b_3 = (1, x, k)$.

This procedure is repeated until the online player either accepts a_t for some $t < k$ or $t = k$. If the online player accepts a_t for some $2 \leq t < k$, the competitive ratio is given by

$$\frac{(t-1)x + t}{(t-1)x + 1}. \tag{26}$$

For $2 \leq t < k$, (26) is increasing in t. Thus, the online player accepts a_1 in the first time period and ends up with a competitive ratio of x, or accepts a_2 and ends up with the abovementioned competitive ratio of $(x+2)/(x+1)$, or accepts b_t for $t = 1, \ldots, k-1$. In the latter case, the online player obviously accepts b_k in the last time period since $x > 1$, and the competitive ratio is given by

$$\frac{(k-1)x + k}{kx}. \tag{27}$$

Set $x := \sqrt{2}$. Then, for $k \geq 4$, (27) is larger than $\sqrt{2}$, i.e.,

$$\frac{(k-1)x+k}{kx} = \frac{k-1}{k} + \frac{1}{\sqrt{2}} \geq \frac{3}{4} + \frac{1}{\sqrt{2}} \approx 1.457 > \sqrt{2},$$

and ${}^{(x+2)}/_{(x+1)} = {}^{(\sqrt{2}+2)}/_{(\sqrt{2}+1)} = \sqrt{2}$. Consequently, for $k \geq 4$, no deterministic algorithm for OKIC with removable items can achieve a competitive ratio smaller than $\sqrt{2}$.

For $k = 3$, set $x := {}^{(1+\sqrt{10})}/_{3} \approx 1.387$. Then, (27) equals x, i.e.,

$$\frac{(k-1)x+k}{kx} = \frac{2}{3} + \frac{3}{1+\sqrt{10}} = \frac{1+\sqrt{10}}{3},$$

and ${}^{(x+2)}/_{(x+1)} \approx 1.418$. Thus, for $k = 3$, no deterministic algorithm for OKIC with removable items can achieve a competitive ratio smaller than ${}^{(1+\sqrt{10})}/_{3}$.
Finally, for $k = 2$, set $x := {}^{(1+\sqrt{17})}/_{4} \approx 1.280$. Then, (27) equals x, i.e.,

$$\frac{(k-1)x+k}{kx} = \frac{1}{2} + \frac{4}{1+\sqrt{17}} = \frac{1+\sqrt{17}}{4}.$$

Therefore, for $k = 2$, no deterministic algorithm for OKIC with removable items can achieve a competitive ratio smaller than ${}^{(1+\sqrt{17})}/_{4}$. □

7 Extension to multiple knapsacks

In this section, we consider the extension of the online knapsack problem with incremental capacity to *multiple knapsacks*. Instead of a single knapsack with increasing capacity, we now have multiple knapsacks, each with increasing capacity: at the beginning, there are m knapsacks and the capacity of each knapsack increases by k units in each time period, as in the case of a single knapsack.

By means of a result by Awerbuch et al. (1996), the algorithms presented in this work can also be used to obtain competitive algorithms for the problem with multiple knapsacks: We run m copies of the single knapsack algorithm ALG, one for each knapsack, denoted by $\text{ALG}_1, \ldots, \text{ALG}_m$. In each time period, the full set of new requests is presented to ALG_1. The set of requests is then reduced by the requests accepted by ALG_1, and passed on to ALG_2, and so forth. If ALG is c-competitive, we obtain a $(c+1)$-competitive algorithm for multiple knapsacks:

Theorem 13 (Awerbuch et al. 1996) *If there exists a c-competitive algorithm* ALG *for* OKIC *with a single knapsack, then there exists a $(c+1)$-competitive algorithm for* OKIC *with m knapsacks.*

The theorem in Awerbuch et al. (1996) is given for a broad class of packing problems, but can be translated directly to the online knapsack problem with incremental capacity. The proof is based on the competitiveness of the single-bin/knapsack algorithm and basic set operations on sets of requests, which also hold if not only one but several requests are presented in each time period.

8 Conclusion and future research

In this paper, we considered the online knapsack problem with incremental capacity. For the restriction to unit weight items, we presented lower bounds on the competitive ratio and algorithms with competitive ratios matching these lower bounds exactly in the randomized case and for $k \to \infty$ in the deterministic case. For limited weights in $\{1, \dots, k\}$, a deterministic algorithm with a competitive ratio matching the lower bound for $k \to \infty$ and a randomized algorithm with a competitive ratio matching the lower bound up to a factor of 3 have been developed. In order to be able to develop algorithms with competitive ratios independent of T, we allowed the online player to remove items and presented a 3-competitive algorithm as well as a lower bound of $\sqrt{2}$. Finally, we show how to obtain competitive algorithms for the problem with multiple knapsacks.

Questions for future research include studying other approaches that might lead to constant competitive ratios such as bounded values, resource augmentation, or fractional packing of items. Moreover, one could study the OKIC in a stochastic setting using average case analysis. For the setting with removable items (where we obtained an online algorithm with constant competitive ratio), a natural question is whether a constant competitive ratio can still be achieved in situations where the onine player has a cost for removing an item from the knapsack in a later time period. Another direction for future research could be to study further generalizations of the online knapsack problem with incremental capacity, e.g., the case where the increase in capacity is not identical in each time period, but varies over time.

Appendix: Proof of Theorem 5

Induction I

Base Case: Eq. (7) holds for $t = 2$:

$$\begin{aligned}\prod_{j=1}^{2-1}\left(1 - p_j\right) &= 1 - p_1\\ &= 1 - \frac{2}{T - 1 + 2}\\ &= \frac{T-1}{T+1}\\ &= \frac{(T-t+1)(T-t+2)}{T(T+1)}.\end{aligned}$$

Inductive Step: Let $t \geq 2,\ t \in \mathbb{N}$, be arbitrary and assume that Eq. (7) holds for t ($\star_1$). Then, Eq. (7) also holds for $t+1$:

$$\begin{aligned}\prod_{j=1}^{(t+1)-1} \left(1 - p_j\right) &= \prod_{j=1}^{t} \left(1 - \frac{2}{T-j+2}\right) \\ &= \prod_{j=1}^{t} \frac{T-j}{T-j+2} \\ &= \prod_{j=1}^{t-1} \frac{T-j}{T-j+2} \cdot \frac{T-t}{T-t+2} \\ &\overset{(\star_1)}{=} \frac{(T-t+1)(T-t+2)}{T(T+1)} \cdot \frac{T-t}{T-t+2} \\ &= \frac{(T-(t+1)+1)\,(T-(t+1)+2)}{T(T+1)}.\end{aligned}$$

Induction II

Base Case: Eq. (8) holds for $i = 2$:

$$\begin{aligned}\prod_{j=1}^{2-1} \left(1 - p_{t-j}\right) &= 1 - p_{t-1} \\ &= 1 - \frac{2}{T-(t-1)+2} \\ &= \frac{T-t+1}{T-t+3} \\ &= \frac{(T-t+1)(T-t+2)}{(T-t+i)(T-t+i+1)}.\end{aligned}$$

Inductive Step: Let $i \geq 2,\ i \in \mathbb{N}$, be arbitrary and assume that Eq. (8) holds for i ($\star_2$). Then, Eq. (8) also holds for $i+1$:

$$\begin{aligned}\prod_{j=1}^{(i+1)-1} \left(1 - p_{t-j}\right) &= \prod_{j=1}^{i} \left(1 - \frac{2}{T-(t-j)+2}\right) \\ &= \prod_{j=1}^{i} \frac{T-t+j}{T-t+j+2} \\ &= \prod_{j=1}^{i-1} \frac{T-t+j}{T-t+j+2} \cdot \frac{T-t+i}{T-t+i+2}\end{aligned}$$

$$\overset{(\star_2)}{=} \frac{(T-t+2)(T-t+1)}{(T-t+i)(T-t+i+1)} \cdot \frac{T-t+i}{T-t+i+2}$$
$$= \frac{(T-t+2)\,(T-t+1)}{(T-t+(i+1))(T-t+(i+1)+1)}.$$

Induction III

Base Case: Eq. (10) holds for $t=2$:

$$\sum_{i=1}^{2-1} \frac{ip_{t-i}}{(T-t+i)(T-t+i+1)} = \frac{p_1}{(T-1)T}$$
$$= \frac{2}{(T-1)T(T+1)}$$
$$= \frac{t(1-t)}{T(T+1)(t-T-1)}.$$

Inductive Step: Let $t \geq 2$, $t \in \mathbb{N}$, be arbitrary and assume that Eq. (10) holds for t ($\star_3$). Then, Eq. (10) also holds for $t+1$:

$$\sum_{i=1}^{(t+1)-1} \frac{ip_{(t+1)-i}}{(T-t-1+i)(T-t-1+i+1)}$$
$$= \sum_{i=1}^{t} \frac{2i}{(T-t-1+i)(T-t+i)(T-t+1+i)}$$
$$= \sum_{i=1}^{t} \frac{2i}{(T'-t+i)(T'-t+1+i)(T'-t+2+i)} \quad \text{with } T' := T-1$$
$$= \sum_{i=1}^{t-1} \frac{2i}{(T'-t+i)(T'-t+i+1)(T'-t+i+2)} + \frac{2t}{T'(T'+1)(T'+2)}$$
$$\overset{(\star_3)}{=} \frac{t(1-t)}{T'(T'+1)(t-T'-1)} + \frac{2t}{T'(T'+1)(T'+2)}$$
$$= \frac{t\left((1-t)(T'+2)+2(t-T'-1)\right)}{T'(T'+1)(T'+2)(t-T'-1)}$$
$$= \frac{t\left(T'-tT'-2T'\right)}{T'(T'+1)(T'+2)(t-T'-1)}$$
$$= \frac{t(-1-t)}{(T'+1)(T'+2)(t-T'-1)}$$
$$= \frac{t(-1-t)}{T(T+1)(t-T)} = \frac{(t+1)(1-(t+1)}{T(T+1)((t+1)-T-1)}.$$

References

Awerbuch B, Azar Y, Fiat A, Leonardi S, Rosén A (1996) On-line competive algorithms for call admission in optical networks. In: Proceedings of the 4th annual European symposium on algorithms (ESA), LNCS, vol 1136, pp 431–444

Babaioff M, Immorlica N, Kempe D (2007) A knapsack secretary problem with applications. In: Proceedings of the 10th international workshop on approximation algorithms for combinatorial optimization (APPROX), pp 16–28

Bienstock D, Sethuraman J, Ye C (2013) Approximation algorithms for the incremental knapsack problem via disjunctive programming. arXiv.1311.4563

Borodin A, El-Yaniv R (1998) Online computation and competitive analysis. Cambridge University Press, Cambridge

Disser Y, Klimm M, Megow N, Stiller S (2014) Packing a knapsack of unknown capacity. In: Proceedings of the 31st international symposium on theoretical aspects of computer science (STACS), pp 276–287

El-Yaniv R, Turpin G, Karp R, Fiat A (1992) Competitive analysis of financial games. In: Proceedings of the 33rd annual IEEE symposium on the foundations of computer science (FOCS), pp 327–333

Han X, Kawase Y, Makino K (2015) Randomized algorithms for online knapsack problems. Theor Comput Sci 562:395–405

Hartline J, Sharp A (2006) An incremental model for combinatorial maximization problems. In: Proceedings of the 5th international workshop on experimental and efficient algorithms (WEA), LNCS, vol 4007, pp 36–48

Ibarra OH, Kim CE (1975) Fast approximation algorithms for the knapsack and sum of subset problems. J ACM 22(4):463–468

Ito H, Kiyoshima S, Yoshidy Y (2012) Constant-time approximation algorithms for the knapsack problem. In: Proceedings of the 9th annual conference on theory and applications of models of computation (TAMC), LNCS, vol 7287, pp 131–142

Iwama K, Taketomi S (2002) Removable online knapsack problems. In: Proceedings of the 29th international colloquium on automata, languages and programming (ICALP), pp 293–305

Iwama K, Zhang G (2003) Removable online knapsack-weighted case. In: Proceedings of the 7th Japan–Korea workshop on algorithms and computation (WAAC), pp 223–227

Iwama K, Zhang G (2007) Optimal resource augmentations for online knapsack. In: Proceedings of the 10th international workshop on approximation algorithms for combinatorial optimization (APPROX), pp 180–188

Karp RM (1972) Reducibility among combinatorial problems. In: Proceedings of a symposium on the complexity of computer computations, pp 85–103 (1972)

Kellerer H, Pferschy U, Pisinger D (2004) Knapsack problems. Springer, New York

Kleywegt AJ, Papastavrou JD (1998) The dynamic and stochastic knapsack problem. Oper Res 46(1):17–35

Lueker GS (1998) Average-case analysis of off-line and on-line knapsack problems. J Algorithms 29(2):277–305

Marchetti-Spaccamela A, Vercellis C (1995) Stochastic on-line knapsack problems. Math Progr 68(1–3):73–104

Martello S, Toth P (1990) Knapsack problems: algorithms and computer implementations. Wiley, Chichester

Megow N, Mestre J (2013) Instance-sensitive robustness guarantees for sequencing with unknown packing and covering constraints. In: Proceedings of the 4th conference on innovations in theoretical computer science (ITCS), pp 495–504

Noga J, Sarbua V (2005) An online partially fractional knapsack problem. In: Proceedings of the 8th international symposium on parallel architectures, algorithms and networks (ISPAN), pp 108–112

Papastavrou JD, Rajagopalan S, Kleywegt AJ (1996) The dynamic and stochastic knapsack problem with deadlines. Manag Sci 42(12):1706–1718

van Slyke R, Young Y (2000) Finite horizon stochastic knapsacks with applications to yield management. Oper Res 48(1):155–172

Yao AC (1977) Probabilistic computations: Toward a unified measure of complexity. In: Proceedings of the 18th annual IEEE symposium on the foundations of computer science (FOCS), pp 222–227

Zhou Y, Chakrabarty D, Lukose R (2008) Budget constrained bidding in keyword auctions and online knapsack problems. In: Proceedings of the 4th workshop on internet and network economics (WINE), pp 566–576