

**MATHEMATICAL ENGINEERING  
TECHNICAL REPORTS**

**Online Minimization Knapsack Problem**

Xin HAN and Kazuhisa MAKINO

METR 2009-04

Feb. 2009

DEPARTMENT OF MATHEMATICAL INFORMATICS  
GRADUATE SCHOOL OF INFORMATION SCIENCE AND TECHNOLOGY  
THE UNIVERSITY OF TOKYO  
BUNKYO-KU, TOKYO 113-8656, JAPAN

**WWW page: <http://www.keisu.t.u-tokyo.ac.jp/research/techrep/index.html>**

The METR technical reports are published as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, notwithstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.

# Online Minimization Knapsack Problem

Xin Han   Kazuhisa Makino

Department of Mathematical Informatics, Graduate School of Information and Technology,  
University of Tokyo, Tokyo, 113-8656, Japan  
hanxin.mail@gmail.com, makino@mist.i.u-tokyo.ac.jp

Feb. 17th, 2009\*

## Abstract

In this paper, we address the online minimization knapsack problem, i. e., the items are given one by one over time and the goal is to minimize the total cost of items that covers a knapsack. We study the *removable* model, where it is allowed to remove old items from the knapsack in order to accept a new item. We obtain the following results.

- (i) We propose an 8-competitive deterministic algorithm for the problem, which contrasts to the result for the online maximization knapsack problem that no online algorithm has a bounded competitive ratio [7].
- (ii) We propose a 2e-competitive randomized algorithm for the problem.
- (iii) We derive a lower bound 2 for deterministic algorithms for the problem.
- (iv) We propose a 1.618-competitive deterministic algorithm for the case in which each item has its size equal to its cost, and show that this is best possible.

## 1 Introduction

Knapsack problem is one of the most classical and studied problems in combinatorial optimization and has a lot of applications in the real world [8]. The (classical) knapsack problem is given a set of items with profits and sizes, and the capacity value of a knapsack, to maximize the total profit of selected items in the knapsack satisfying the capacity constraint. This problem is also called the *maximization* knapsack problem (Max-Knapsack). Many kinds of variants and generalizations of the knapsack problem have been investigated so far [8]. Among them, the *minimization* knapsack problem (Min-Knapsack) is one of the most natural ones (see [1, 2, 3, 4] and [8, pp. 412-413]), that is given a set of items associated with costs and sizes, and the size of a knapsack, to minimize the total cost of selected items that cover the knapsack. Note that Min-Knapsack can be transformed into Max-Knapsack in polynomial time (and vice versa), i.e., they are polynomially equivalent. However, Min-Knapsack and Max-Knapsack exhibit relevant differences in approximation factors for the algorithms. For example, a polynomial time approximation scheme (PTAS) for Max-Knapsack does not directly lead to a PTAS for Min-Knapsack.

---

\*Original version written on Jan. 30th, 2009

In this paper, we focus on the online version of problem Min-Knapsack. To our best knowledge, this is the first paper on online minimization knapsack problem. Here, “online” means that items are given over time, i.e., after a decision of rejection or acceptance is made on the current item, the next item is given, and once an item is rejected or removed, it cannot be considered again. The goal of the online minimization knapsack problem is the same as the offline version, i.e., to minimize the total cost.

**Related work:** It is well-known that offline Max-Knapsack and Min-Knapsack both admit a fully polynomial time approximation scheme (FPTAS) [1, 4, 8]. As for the online maximization knapsack problem, it was first studied on average case analysis by Marchetti-Spaccamela and Vercellis [11]. They proposed a linear time approximation algorithm such that the expected difference between the optimal and the approximation solution value is  $O(\log^{3/2} n)$  under the condition that the capacity of the knapsack grows proportionally to  $n$ , the number of items. Lueker [10] further improved the expected difference to  $O(\log n)$  under a fairly general condition on the distribution. Recently, Iwama and Taketomi [6] studied the problem on worst case analysis. They obtained a 1.618-competitive algorithm for the online Max-Knapsack under the *removable* condition, if each item has its size equal to its profit. Here the *removable* condition means that it is allowed to remove some items in the knapsack in order to accept a new item. They also showed that this is best possible by providing a lower bound 1.618 for this case. For the general case, Iwama and Zhang [7] showed that no algorithm for online Max-Knapsack has a bounded competitive ratio, even if the removal condition is allowed. Some generalizations of the online Max-Knapsack such as resource augmentations were also investigated [7, 13].

**Our results:** In this paper, we study the online minimization knapsack problem. We first show that no algorithm has a bounded competitive ratio, if the *removable* condition is not allowed. Under the removable condition, we propose two *deterministic* algorithms for the online Min-Knapsack. The first one is simple and has competitive ratio  $\Theta(\log \Delta)$ , where  $\Delta$  is the ratio of the maximum size to the minimum size in the items, and the second one has competitive ratio 8. This constant-competitive result for the online Min-Knapsack contracts to the result for the online Max-Knapsack that no online algorithm has a bounded competitive ratio [7], which is surprising, since problems Max-Knapsack and Min-Knapsack are expected to have the same behavior from a complexity viewpoint (see Table 1).

The first algorithm is motivated by the observation: if all the items have the same size, then a simple greedy algorithm (called *Lowest Cost First* strategy) of picking items with the lowest cost first provides an optimal solution. The algorithm partitions the item set into  $\lceil \log \Delta \rceil + 1$  subsets  $F_j$  by their *size*. When a new item  $d_t$  is given, the algorithm guesses the optimal value within  $O(1)$  approximation factor, by using only the items in the knapsack together with the new item  $d_t$ , and for each class  $F_j$ , chooses items by Lowest Cost First strategy. Since each class  $F_j$  has cost at most  $O(1)$  times the optimal value, we have an  $O(\log \Delta)$ -competitive algorithm, where we also provide a lower bound of the algorithm to show that it is  $\Theta(\log \Delta)$ -competitive.

Note that the first algorithm keeps too many extra items in the knapsack to guess the optimal value of the Min-Knapsack. In order to obtain an  $O(1)$ -competitive online algorithm, we again represent the knapsack as the union of subsets  $F_j$  ( $j \geq 0$ ). Here classes  $F_j$ 's are defined by their *cost* rather than their size. We then guess the optimal value by solving fractional Max-Knapsack

problems for classes  $F_j$ . Since each class  $F_j$  has cost at most  $2^{2-j}$  times the optimal value, we have an 8-competitive algorithm.

We also show that no deterministic online algorithm achieves competitive ratio less than 2, and provides a *randomized* online algorithm with competitive ratio  $2e \approx 5.44$ . We finally consider the case in which each item has its cost equal to its size. Similarly to the Max-Knapsack problem, we show that the online Min-Knapsack problem admits 1.618-competitive algorithm which matches the lower bound.

Table 1 summarizes the current status of the complexity of problems Max-Knapsack and Min-Knapsack, where the bold letters represent the results obtained in this paper.

Table 1: The current status of the complexity of problems Max-Knapsack and Min-Knapsack

			Max-Knapsack		Min-Knapsack	
			lower bound	upper bound	lower bound	upper bound
offline			FPTAS [5]		FPTAS [1]	
online	non-removable	general case	unbounded [6]		<b>unbounded</b>	
		size = cost	unbounded [6]		<b>unbounded</b>	
	removable	general case	unbounded [7]		<b>2</b>	<b>8</b>
		size = cost	1.618 [6]	1.618 [6]	<b>1.618</b>	<b>1.618</b>

The rest of the paper is organized as follows. Section 2 gives definitions of the online Min-Knapsack problem, and show that the “removable” condition is necessary for the online Min-Knapsack problem. Section 3 presents algorithms for the online Min-Knapsack problem, and Section 4 gives a lower bound 2 for the online Min-Knapsack problem. Finally, in Section 5, we consider the case where each item has its cost equal to its size.

## 2 Preliminaries

In this section, we first give the definition of the online Min-Knapsack problem and show that why the removable condition is necessary for the problem.

Let us first define the offline minimization knapsack problem.

Problem Min-Knapsack

Input: A set of items  $D = \{d_1, \dots, d_n\}$  associated with cost  $c : D \rightarrow \mathbb{R}_+$  and size  $s : D \rightarrow \mathbb{R}_+$ .

Output: A set of items  $F \subseteq D$  that minimizes  $\sum_{f \in F} c(f)$  subject to  $\sum_{f \in F} s(f) \geq 1$ .

Here we assume w.l.o.g. that the size of the knapsack is 1. For a set  $U \subseteq D$ , let  $c(U) = \sum_{u \in U} c(u)$  and  $s(U) = \sum_{u \in U} s(u)$ .

In the online model, the objective is the same with the offline version. But the input is given over time. Namely, the knapsack of size 1 is known beforehand, and after a decision is made on

the current item  $d_t$  associated with  $c(d_t)$  and  $s(d_t)$ , the next one  $d_{t+1}$  is given. Once items are discarded, they cannot be used again, even for estimating an optimal value of the problem. We analyze online algorithms by using one of the standards: the competitive ratio. Given an input sequence  $L$  and an online algorithm  $A$ , the *competitive ratio* of algorithm  $A$  is defined as follows:

$$R_A = \sup_L \frac{A(L)}{OPT(L)},$$

where  $OPT(L)$  and  $A(L)$  denotes the costs obtained by an optimal algorithm and the algorithm  $A$ , respectively. If  $A(L)$  has no feasible solution, then we define  $A(L) = +\infty$ . If  $A$  is a randomized algorithm, then we have  $R_A = \sup_L \frac{E[A(L)]}{OPT(L)}$ .

In this paper, we consider *removable condition* for the online Min-Knapsack, i.e., it is allowed to remove or discard old items in the knapsack. It follows from the following lemma that removable condition is necessary to have a bounded competitive ratio.

**Lemma 1** *If at least one of the following conditions is not satisfied, then no algorithm has a bounded competitive ratio for the online Min-Knapsack problem.*

- (i) *Before the total size of the items given so far is smaller than 1 (the size of the knapsack), no item is rejected.*
- (ii) *It is allowed to remove old items in the knapsack when a new item is given.*

*Proof.* For (i), let us see the following instance. Given two items  $d_1$  and  $d_2$  with size 0.9 and 0.1, respectively. If an online algorithm does not accept both items, then the adversary stops the input after the second round. The online algorithm is infeasible, while the optimal algorithm does have a feasible solution. Thus, the competitive ratio of the online algorithm is unbounded.

For (ii), let us see the following input instance. The first item  $d_1$  has its size  $M$  and cost  $M$ , where  $M$  is arbitrarily large. Note that by (i) any online algorithm has to accept the item and keep it in the knapsack. After keeping the first one in the knapsack, the second item  $d_2$  with size 1 and cost 1 is given. If it is not allowed to remove the first item, the online algorithm has a solution whose value is at least  $M$ . However, the optimal value is at most 1 by keeping the second item. When  $M$  goes to infinity, the competitive ratio is unbounded.  $\square$

From Lemma 1, in the subsequential sections, we consider the online Min-Knapsack problem under the removable condition.

### 3 Algorithms for the general case

In this section, we present algorithms for the online Min-Knapsack problem under the removable condition. Note that in our model, once an item is removed or rejected, it cannot be used again, even for estimating the optimal value. Therefore, we have to keep items in the knapsack so that they adjust any forthcoming input sequence.

To construct an online algorithm with small competitive ratio, there are two points that we have to keep in mind: (I) keep feasible any time (i.e., the total size in the knapsack is at least 1), after the total size of the items given so far is at least 1, and (II) the total cost in the knapsack is not too far from the optimal cost, where (I) follows from (i) in Lemma 1.

### 3.1 A simple deterministic algorithm

In this subsection, we give a simple online algorithm with a competitive ratio  $\Theta(\log \Delta)$ , where  $\Delta$  is the ratio of the maximum size to the minimum size. The online algorithm is motivated by the observation: if all the items have the same size, then the greedy algorithm (*Lowest Cost First* selection strategy) of picking items with the lowest cost first provides an optimal solution.

For a nonnegative integer  $t$ , let  $D(t)$  denote the set of the first  $t$  items, i.e.,  $D(t) = \{d_1, \dots, d_t\}$ , and let  $F(t)$  denote the set of items that our algorithm keeps in the knapsack after the  $t$ -round. Let  $t_0$  be the first time when there is a feasible solution for  $D(t)$ , i.e.,  $t_0 = \min\{t \mid \sum_{i=1}^t s(d_i) \geq 1\}$ . By Lemma 1 (i), for  $t < t_0$ , our algorithm keeps all the items, i.e.,  $F(t) = D(t)$ .

Let us then consider when  $t \geq t_0$ . For an integer  $-\infty < j < +\infty$ , define

$$S_j = \{d \in D \mid 2^j < s(d) \leq 2^{j+1}\},$$

$$D_j(t) = D(t) \cap S_j \text{ and } F_j(t) = F(t) \cap S_j.$$

Our algorithm keeps  $F(t)$  as the union of  $\lceil \log_2 \Delta + 1 \rceil$  classes  $F_j(t)$ . When a new item  $d_t$  is given, the algorithm computes a guessed value  $\beta(t)$  for the optimal cost  $OPT(D(t))$  for the input  $D(t)$  such that  $\beta(t) = O(1)OPT(D(t))$ , by using only the items in the knapsack  $F(t-1)$  together with the new item  $d_t$ , and then for each  $j$ , we construct  $F_j(t)$  from  $F_j(t-1)$  by keeping the items with the total cost at most  $3\beta(t)$  by the Lowest Cost First strategy.

Formally, the algorithm when  $t \geq t_0$  is described as follows.

Algorithm A
<ol style="list-style-type: none"> <li>1. <math>E(t) := F(t-1) \cup \{d_t\}</math>.</li> <li>2. Guess: Compute a value <math>\alpha(t)</math> by an approximation algorithm (e.g., [1, 2]) with <math>E(t)</math> as the input. Set <math>\beta(t) := \min\{\beta(t-1), \alpha(t)\}</math>.</li> <li>3. For each <math>j</math>, <math>F_j(t) := E(t) \cap S_j</math> and if <math>c(F_j(t)) &gt; 3\beta(t)</math> then repeatedly remove an item with the highest cost until <math>c(F_j(t)) \leq 3\beta(t)</math>.</li> <li>4. <math>F(t) := \bigcup_j F_j(t)</math></li> </ol>

Note that, for any time  $t$ , the total number of classes  $F_j(t)$  needed in the algorithm is bounded by  $\lceil \log_2 \Delta + 1 \rceil$ . Therefore, the algorithm is  $O(\log \Delta)$ -competitive, if we have  $s(F(t)) \geq 1$  (i.e.,  $F(t)$  is feasible) and  $\beta(t) = O(1)OPT(D(t))$  for all  $t \geq t_0$ . We shall show them by a series of lemmas.

**Lemma 2** *Let  $j$  be an integer. At time  $t \geq t_0$ , we have  $c(p) \geq c(q)$  for all  $p \in D_j(t) - F_j(t)$  and  $q \in F_j(t)$ .*

*Proof.* For a fixed  $j$ , we prove this lemma by induction. It is easy to see that the lemma holds for time  $t = t_0$ , since Step 3 repeatedly removes an item with the highest cost until  $c(F_j(t)) \leq 3\beta(t)$ .

Assume that the lemma holds for time  $t_1 \geq t_0$  and consider time  $t = t_1 + 1$ . If  $d_t \notin F_j(t)$ , then the inductive hypothesis implies that the lemma holds, since Step 3 always removes the item with the highest cost. If  $d_t \in F_j(t)$ , let  $\gamma$  be the cost of a cheapest item in  $D_j(t) - F_j(t)$ . We show that

$c(d_t) \leq \gamma$ , which together with the inductive hypothesis implies that the lemma is true for this case.

To prove  $c(d_t) \leq \gamma$ , let  $t' (\leq t)$  be the maximum time when an item with cost  $\gamma$  is discarded. Since all the items in  $F_j(t')$  have costs at most  $\gamma$ , from the definitions of  $\gamma$  and  $t'$ , we have

$$F_j(t') \subseteq F_j(t).$$

Thus, if  $c(d_t) > \gamma$ , then  $t' < t$  holds by  $d_t \in F_j(t)$ , and the monotonicity of  $\beta$ , we have

$$c(F_j(t)) \geq c(F_j(t')) + \gamma > 3\beta(t') \geq 3\beta(t),$$

which is a contradiction.  $\square$

Let  $F^*(t)$  denote an optimal solution for an input  $D(t)$  and  $F_j^*(t) = F^*(t) \cap S_j$ .

**Lemma 3** *For a time  $t \geq t_0$ , assume that there is a feasible solution in  $F(t)$ , i.e.,  $s(F(t)) \geq 1$ . Then, for all  $j$ , we have  $c(F_j(t)) \geq 2\beta(t)$  if  $F_j^*(t) \not\subseteq F_j(t)$ .*

*Proof.* Assume that  $F_j^*(t) \not\subseteq F_j(t)$  for some  $j$  and  $t \geq t_0$ . Let  $\gamma$  be the cost of a cheapest item in  $D_j(t) - F_j(t)$ , and let  $t' (\leq t)$  be the maximum time when an item with cost  $\gamma$  is discarded. As mentioned in the proof of Lemma 2, we have  $F_j(t') \subseteq F_j(t)$ . This implies  $c(F_j(t')) \leq c(F_j(t))$ . Since  $c$  is nonnegative and  $F_j^*(t) \not\subseteq F_j(t)$ , we also have

$$\gamma \leq c(F_j^*(t)) \leq OPT(D(t)) \leq \beta(t),$$

where the last inequality holds from the assumption. This implies  $c(F_j(t')) \geq 3\beta(t') - \beta(t) \geq 2\beta(t)$ , and hence  $c(F_j(t)) \geq 2\beta(t)$ .  $\square$

**Lemma 4** *At any time  $t \geq t_0$ ,  $F(t)$  contains a feasible solution for  $D(t)$  with cost at most  $2OPT(D(t))$ .*

*Proof.* Remember that  $F^*(t)$  denotes an optimal solution for an input  $D(t)$  and  $F_j^*(t) = F^*(t) \cap S_j$ . For each  $j$ , we show that there exists  $G_j$  in  $F_j(t)$  such that

$$s(G_j) \geq s(F_j^*(t)) \quad \text{and} \quad c(G_j) \leq 2c(F_j^*(t)).$$

This completes the proof, since  $\bigcup_j G_j$  is a feasible solution for  $D(t)$  with cost at most  $2OPT(D(t))$ .

Let us construct  $G_j$  as follows.

If  $F_j^*(t) \subseteq F_j(t)$ , then  $G_j := F_j^*(t)$ . Otherwise, we construct  $G_j$  from  $F_j^*(t) \cap F_j(t)$  by adding the cheapest  $2|F_j^*(t) \setminus F_j(t)|$  items from  $F_j(t) \setminus F_j^*(t)$ .

Since by Lemma 2  $c(p) \geq c(q)$  holds for all  $p \in D_j(t) - F_j(t)$  and  $q \in F_j(t)$ , we have

$$c(G_j) \leq 2c(F_j^*(t)).$$

By the definition of  $S_j$ , we have  $s(p)/2 \leq s(q)$  for any items  $p$  and  $q$  in  $S_j$ , which implies  $s(G_j) \geq s(F_j^*(t))$ . These prove the lemma if we can show  $|F_j(t) \setminus F_j^*(t)| \geq 2|F_j^*(t) \setminus F_j(t)|$ .

We prove this by induction on time  $t \geq t_0$ . At time  $t = t_0$ , we have  $\beta(t) \geq OPT(D(t))$  since before  $t_0$  all the items are accepted in the knapsack. If  $F_j^*(t) \not\subseteq F_j(t)$ , by Lemma 3, we have

$$c(F_j(t)) \geq 2\beta(t) \geq 2OPT(D(t)) \geq 2c(F_j^*(t)),$$

and hence we have

$$c(F_j(t) \setminus F_j^*(t)) \geq 2c(F_j^*(t) \setminus F_j(t)).$$

By Lemma 2, we have  $c(p) \leq c(q)$  for any two items  $p \in F_j(t) \setminus F_j^*(t)$  and  $q \in F_j^*(t) \setminus F_j(t)$ . This implies that the lemma is true for  $t = t_0$ .

Assume that  $|F_j(t) \setminus F_j^*(t)| \geq 2|F_j^*(t) \setminus F_j(t)|$  holds for  $t = t_1 (\geq t_0)$  and consider time  $t = t_1 + 1$ . By the inductive hypothesis,  $F(t_1)$  contains a feasible solution for  $D(t_1)$ . Thus we have  $\beta(t) \geq OPT(D(t))$ . By Lemma 3, we have

$$c(F_j(t)) \geq 2\beta(t) \geq 2OPT(D(t)) \geq 2c(F_j^*(t)).$$

Then by applying the same argument for  $t = t_0$  to this case, we can prove that  $|F_j(t) \setminus F_j^*(t)| \geq 2|F_j^*(t) \setminus F_j(t)|$ .  $\square$

**Lemma 5** *Algorithm A is  $O(\log \Delta)$ -competitive.*

*Proof.* By Lemma 4,  $F(t)$  contains a feasible solution for  $D(t)$  with cost at most  $2OPT(D(t))$ .

Since (offline) Min-Knapsack problem admits a FPTAS [1],

$$\beta(t) \leq (1 + \epsilon)OPT(F(t)) \leq 2(1 + \epsilon)OPT(D(t))$$

for some  $\epsilon > 0$  and the cost by algorithm A satisfies

$$A(D(t)) \leq 3(\lceil \log_2 \Delta \rceil + 1)\beta(t),$$

and hence we have  $A(D(t)) \leq O(\log \Delta)OPT(D(t))$ .  $\square$

The next lemma shows that the analysis of the competitive ratio for algorithm A is tight.

**Lemma 6** *Algorithm A is  $\Omega(\log \Delta)$ -competitive.*

*Proof.* To prove this lemma, we present an instance  $D$  such that  $A(D) \geq \log \Delta \cdot OPT(D)$ .

For  $0 \leq i \leq k$ , let  $b_i$  be an item with  $s(b_i) = c(b_i) = 2^{-i}$ , and we construct an input sequence  $D$  by  $D = D^{(0)}, D^{(1)}, \dots, D^{(k)}$ , where  $D^{(i)}$  is a sequence consisting of  $2^i$   $b_i$ 's. Note that this instance has an optimal solution  $F^* = \{b_0\}$  whose cost is  $OPT(D) = 1$ . On the other hand, algorithm A keeps all the items, and hence  $A(D) = k + 1 > \log_2 \Delta \cdot OPT(D)$ , where  $\Delta = 2^k$  is the ratio of the largest size to the smallest size.  $\square$

By Lemmas 5 and 6, we have the following theorem.

**Theorem 1** *Algorithm A is  $\Theta(\log \Delta)$ -competitive.*

### 3.2 An improved deterministic algorithm

Note that the first algorithm keeps too many extra items in the knapsack to keep a feasible solution and to guess the optimal value of the Min-Knapsack. In order to obtain an  $O(1)$ -competitive online algorithm, for any time  $t (\geq t_0)$ , we represent the knapsack  $F(t)$  as the union of subsets  $F_j(t)$  ( $j \geq 0$ ) which satisfy the following three conditions.

- A guessed value  $\beta(t)$  satisfies  $\beta(t) \leq r \cdot OPT(D(t))$  for some constant  $r > 1$ .
- For each  $j \geq 0$ ,  $c(F_j(t)) \leq 2\beta(t)/r^j$ .
- $F(t) := \bigcup_j F_j(t)$  satisfies the feasibility, i.e.,  $s(F(t)) \geq 1$ .

It is not difficult to see that the algorithm has constant competitive ratio if it satisfies all the conditions above. We now show how to construct such  $F_j$ 's.

Let  $F(t-1)$  denotes a set of items in the knapsack at time  $t-1$ , and let  $E(t) := F(t-1) \cup \{d_t\}$ . For a guessed value  $\beta(t)$ , let  $E_j(t) = \{d_i \in E(t) \mid c(d_i) \leq \beta(t)/r^j\}$  and construct  $F_j(t)$  from  $E_j(t)$  by repeatedly removing an item  $e$  with the highest unit cost  $\frac{c(e)}{s(e)}$ , until the total cost becomes at most  $2\beta(t)/r^j$ . Clearly this construction assures the second condition above.

To assure the first and third conditions, we first initialize  $\beta(t)$  by  $\beta(t) := c(E(t))$  if  $t = t_0$ ; otherwise  $\beta(t) := \beta(t-1)$ . We check if  $s(F_j(t)) \geq 1$  for each  $j$ . Let  $\ell$  be the maximum number  $j$  such that  $s(F_j(t)) \geq 1$ . Then we have  $OPT(D(t)) \leq c(F_j(t)) \leq 2\beta(t)/r^\ell$ . If  $OPT(D(t)) > 2\beta(t)/r^{\ell+1}$  holds in addition, then  $2\beta(t)/r^\ell$  is a good guessed value for  $OPT(D(t))$ . However, in general this is not true, since some item in  $D(t)$  has been already discard before round  $t$ , and hence  $2\beta(t)/r^\ell$  may not be a good guessed value for  $OPT(D(t))$ . In order to overcome this difficulty, we solve the following (offline) fractional maximization knapsack problem for each class  $F_j(t)$ .

$$\begin{aligned}
& \max && \sum_{f \in F_j(t)} s(f) \cdot x(f) \\
& \text{s.t.} && \sum_{f \in F_j(t)} c(f) \cdot x(f) \leq \beta(t)/r^j; \\
& && 0 \leq x(f) \leq 1, \quad f \in F_j(t).
\end{aligned} \tag{1}$$

Let  $FKP(F_j(t), \beta(t)/r^j)$  denote the optimal value of (1), where the second argument  $\beta(t)/r^j$  denotes the capacity of the knapsack. It is well-known [8] that the fractional knapsack problem can be solved by a greedy approach for  $s(f)/c(f)$ . Let  $\ell = \max\{j \mid FKP(F_j(t), \beta(t)/r^j) \geq 1\}$ . Then we can see below that  $\beta(t)/r^\ell$  is a good guessed value and  $F_\ell(t)$  is feasible for our problem.

Formally, the algorithm is described as follows.

Algorithm B for  $t \geq t_0$

1.  $E(t) := F(t-1) \cup \{d_t\}$ . If  $t = t_0$ , then  $\alpha(t) := c(E(t))$ ; otherwise  $\alpha(t) := \beta(t-1)$ .
2. For each integer  $j \geq 0$ , construct a class  $F_j(t)$  as follows.
  - 2.1 Let  $E_j(t) := \{d_i \in E(t) \mid c(d_i) \leq \alpha(t)/r^j\}$  where  $E_j(t)$  is not constructed if  $E_j(t) = \emptyset$ .
  - 2.2 Construct  $F_j(t)$  from  $E_j(t)$  by repeatedly removing an item  $e$  with the highest unit cost  $\frac{c(e)}{s(e)}$ , until the total cost becomes at most  $2\alpha(t)/r^j$ .
3. Let  $\ell = \max\{j \mid FKP(F_j(t), \alpha(t)/r^j) \geq 1\}$ , let  $F(t) := \bigcup_{j \geq \ell} F_j(t)$  and  $\beta(t) := \alpha(t)/r^\ell$ .

Observe that in Step 2.1 of the algorithm  $E_j(t)$  is empty for all  $j$  with  $\alpha(t)/r^j < \min\{c(d) \mid d \in D(t)\}$ , and we have  $\alpha(t) \leq t \max\{c(d) \mid d \in D(t)\}$ . Hence, the number of nonempty  $E_j(t)$  is bounded by  $O(\log \frac{t \max\{c(d) \mid d \in D(t)\}}{\min\{c(d) \mid d \in D(t)\}})$ .

**Lemma 7** *At any time  $t \geq t_0$ , the index  $\ell$  in Step 3 must exist.*

*Proof.* We prove this lemma by induction on  $t$ . When  $t = t_0$ , we have  $\alpha(t) = c(E(t)) (= c(D(t)))$  and  $E_0(t) = D(t)$ . After Step 2.2, we have  $F_0(t) = E_0(t)$ , since  $c(E_0(t)) = \alpha(t) < 2\alpha(t)$ . We also have  $FKP(F_0(t), \alpha(t)) = s(D(t)) \geq 1$ , where the last inequality follows from the definition of  $t_0$ . Therefore, the lemma holds for  $t = t_0$ .

Assume that the lemma holds for time  $t = t_1 (\geq t_0)$ , i.e.,  $FKP(F_0(t_1), \alpha(t_1)) \geq 1$ , and consider time  $t = t_1 + 1$ .

At time  $t$ , if the new item  $d_t$  is not selected in  $F_0(t)$  at Step 2.2, then we have  $F_0(t) = F_0(t_1)$ . Then by the inductive hypothesis, we have  $FKP(F_0(t), \alpha(t)) \geq 1$ , where we note that  $\alpha(t) = \beta(t_1)$ . On the other hand if  $d_t$  is selected in  $F_0(t)$  at Step 2.2, we have

$$FKP(F_0(t), \alpha(t)) = FKP(F_0(t), \beta(t_1)) \geq FKP(F_0(t_1), \beta(t_1)) \geq 1,$$

where the first inequality  $FKP(F_0(t), \beta(t_1)) \geq FKP(F_0(t_1), \beta(t_1))$  follows from the greedy construction of  $F_0(t)$  from  $E_0(t)$ . Hence the lemma holds for  $t = t_1 + 1$ .  $\square$

For a  $U \subseteq D$  and a positive integer  $p$ , let  $KP(U, p)$  denote the optimal value for the knapsack problem that maximize  $\sum_{u \in U} s(u) \cdot x(u)$  subject to  $\sum_{u \in U} c(u) \cdot x(u) \leq p$  and  $x(u) \in \{0, 1\}$  for all  $u \in U$ . By definition, we have  $KP(U, p) \leq FKP(U, p)$ .

**Lemma 8** *At any time  $t (\geq t_0)$ , we have  $FKP(F_j(t), \alpha(t)/r^j) \geq KP(D(t), \alpha(t)/r^j)$  for all  $j \geq 0$ .*

*Proof.* At time  $t (\geq t_0)$ , let  $D_j(t)$  denote the set of items with cost at most  $\alpha(t)/r^j$  in  $D(t)$ . We shall prove that

$$FKP\left(F_j(t), \frac{\alpha(t)}{r^j}\right) = FKP\left(D_j(t), \frac{\alpha(t)}{r^j}\right).$$

Observe that: i)  $\beta(t)$  and  $\alpha(t)$  are non-increasing functions, ii) if  $D_j(t) \geq \alpha(t)/r^j$  then the total cost of  $F_j(t)$  is at least  $\alpha(t)/r^j (= 2\alpha(t)/r^r - \alpha(t)/r^j)$ , since every item in  $F_j(t)$  has cost at most

$\alpha(t)/r^j$ , and iii)  $s(p)/c(p) \leq s(q)/c(q)$  holds for any items  $p \in (D_j(t) - F_j(t))$  and  $q \in F_j(t)$  such that  $q$  is contained in an optimal solution of  $FKP\left(F_j(t), \frac{\alpha(t)}{r^j}\right)$ , by the greedy construction of  $F_j(t)$  in Step 2. Therefore, we have

$$FKP\left(F_j(t), \frac{\alpha(t)}{r^j}\right) = FKP\left(D_j(t), \frac{\alpha(t)}{r^j}\right).$$

This implies

$$FKP\left(F_j(t), \frac{\alpha(t)}{r^j}\right) = FKP\left(D_j(t), \frac{\alpha(t)}{r^j}\right) \geq KP\left(D_j(t), \frac{\alpha(t)}{r^j}\right) = KP\left(D(t), \frac{\alpha(t)}{r^j}\right).$$

□

**Lemma 9** *At any time  $t (\geq t_0)$   $\beta(t)$  satisfies  $\beta(t) < r \cdot OPT(D(t))$ .*

*Proof.* Assume this lemma does not hold, i.e.,  $\beta(t) \geq r \cdot OPT(D(t))$ . Then we have

$$FKP(F_1(t), \beta(t)/r) \geq KP(D(t), \beta(t)/r) \geq \overline{KP}(D(t), OPT(D(t))) \geq 1,$$

where the first inequality follows from Lemma 8, the second one follows from assumption  $\beta(t) \geq r \cdot OPT(D(t))$ , and the last one holds for  $t \geq t_0$ . This contradicts the maximality of  $\ell$  at Step 3. □

**Theorem 2** *Algorithm B is 8-competitive, i.e.,  $B(D(t)) \leq 8OPT(D(t))$  for any  $t \geq t_0$ .*

*Proof.* By Lemma 7 and the definition of  $F_0(t)$ , we have

$$s(F_0(t)) \geq FKP(F_0(t), \beta(t)) \geq 1,$$

i.e.,  $F_0(t)$  is a feasible solution for the Min-Knapsack with the input  $D(t)$ , and hence  $F(t) = \bigcup_{j \geq 0} F_j(t)$  is also feasible. The total cost in the knapsack  $F(t)$  satisfies

$$c(F(t)) \leq 2\beta(t) \sum_{i=0}^{\infty} r^{-i} < \frac{2r\beta(t)}{(r-1)} \leq \frac{2r^2}{r-1} OPT(D(t)),$$

where the last inequality follows from Lemma 9. Since  $\frac{2r^2}{r-1} = 8$  if  $r = 2$ , this completes the proof. □

### 3.3 A randomized algorithm

Observe that the worst case of algorithm B is when the optimal cost  $OPT(D(t))$  is sufficiently close to  $\beta(t)/r$ . We find that a randomized technique in [9] can foil the worst case. Namely, let  $\xi$  be a random variable uniformly distributed in  $[0, 1)$ . Then the competitive ratio can be improved if algorithm B uses  $\alpha(t_0) = r^\xi c(E(t_0))$  instead of  $\alpha(t_0) = c(E(t_0))$  in Step 1, i.e., if we shift  $\alpha(t_0)$  by multiplying a factor  $r^\xi$ . Let us call this randomized algorithm RB.

We shall below show that  $E[RB(D)]/OPT(D) \leq 2e$  for all  $D$ , where we deal with the *oblivious* online randomized algorithms [12].

**Theorem 3** *For  $r = e$ , algorithm RB is  $2e$ -competitive.*

*Proof.* For any  $t \geq t_0$ , we note that the guessed value  $\beta(t)$  and the cost  $RB(D(t))$  by algorithm RB are random variables, while  $OPT(D(t))$  is not. For simplicity, in this proof we write  $OPT$  instead of  $OPT(D(t))$ . Let us compare  $\beta(t)$  with  $OPT$ . There are two cases:  $\beta(t) < OPT$  and  $\beta(t) \geq OPT$ . We first prove that  $E[\beta(t) \mid \beta(t) \geq OPT] \leq (e - 1)OPT$ . We rewrite  $\beta(t)$  and  $OPT$  as

$$OPT = e^{\ln OPT} \quad \text{and} \quad \beta(t) = e^{\xi+i+\delta},$$

where  $i$  is an integer and  $0 \leq \delta < 1$ . By Lemma 9 and by assumption  $\beta(t) \geq OPT$ , we have

$$1 \leq \beta_t/OPT < e (= e^1), \quad \Rightarrow 0 \leq \xi + i + \delta - \ln OPT < 1.$$

Since  $\xi$  is uniformly distributed in  $[0, 1)$ , the random variable  $x = \xi + i + \delta - \ln OPT$  is also uniformly distributed in  $[0, 1)$ . Thus, we have

$$E\left[\frac{\beta(t)}{OPT} \mid \beta(t) \geq OPT\right] = \int_0^1 e^x dx = e - 1.$$

Since  $OPT$  is a fixed number, we have  $E[\frac{\beta(t)}{OPT} \mid \beta(t) \geq OPT] = E[\beta(t) \mid \beta(t) \geq OPT]/OPT$ . Hence we obtain  $E[\beta(t) \mid \beta(t) \geq OPT] = (e - 1)OPT$ .

Define

$$\Pr[\beta(t) < OPT] = p,$$

where  $0 \leq p \leq 1$ . Then we have

$$\begin{aligned} E[RB(D(t))] &= E\left[\sum_{j=0}^{+\infty} 2\beta(t)e^{-j}\right] \leq \frac{2e}{e-1}E[\beta(t)] \\ &\leq \frac{2e}{e-1}\left(pE[\beta(t) \mid \beta(t) < OPT] + (1-p)E[\beta(t) \mid \beta(t) \geq OPT]\right) \\ &\leq \frac{2e}{e-1}\left(pOPT + (1-p)(e-1)OPT\right) \leq 2e \cdot OPT. \end{aligned}$$

□

## 4 A lower bound on the online Min-Knapsack problem

In this section, we give a lower bound 2 for the competitive ratio for the online Min-Knapsack problem. The main idea of our proof is given as follows. Assume that an online algorithm has competitive ratio smaller than 2. After  $t \geq t_0$ , if a small item with a small cost is given, the algorithm has to accept it, since otherwise the adversary can kill the algorithm by giving an item with large size and zero cost, i.e., the adversary will cause the online algorithm to have competitive ratio at least 2. However, after accepting small items, the total cost in the knapsack would be arbitrarily close to twice the total cost before accepting small items, This implies that the competitive ratio is at least 2.

**Theorem 4** *Any deterministic algorithm for the online Min-Knapsack problem has competitive ratio at least 2.*

*Proof.* Assume an online algorithm  $A$  has competitive ratio  $r < 2$ . Let us represent an item  $d_t$  by  $d_t = (s(d_t), c(d_t))$ .

The first item  $d_1 = (1, 1)$  is given, by Lemma 1, algorithm  $A$  accepts it. We have a feasible solution with cost 1 in the knapsack at time  $t = 0$ . We then consider the sequence of items

$$(1/2^2, 1/2), (1/2^3, 1/2^2), \dots, (1/2^{k+1}, 1/2^k).$$

If the second item  $(1/2^2, 1/2)$  is given, then algorithm  $A$  has to accept it, since otherwise the adversary gives the third item  $(1 - 1/2^2 (= 3/4), 0)$  and stop the input sequence. In this case, algorithm  $A$  has a feasible solution with cost 1 while the optimal cost is  $1/2$ . This contradicts the assumption  $r < 2$ .

After algorithm  $A$  accepts the second item  $(1/2^2, 1/2)$ , the item  $(1/2^3, 1/2^2)$  is given. Then algorithm  $A$  has to keep all the items given so far, i.e.,  $(1, 1)$ ,  $(1/2^2, 1/2)$ , and  $(1/2^3, 1/2^2)$ .  $(1, 1)$  is kept by Lemma 1. If  $(1/2^3, 1/2^2)$  is not accepted, then the adversary will give the next item  $(1 - 1/2^3 (= 7/8), 0)$  and stop the input sequence. Then algorithm  $A$  would have the competitive ratio at least 2, which is a contradiction. On the other hand, if  $(1/2^2, 1/2)$  is removed, then the adversary will give the next item  $(1 - 1/2^2 (= 3/4), 0)$  and stop the input. In this case, since the two items with sizes  $1/8$  and  $3/4$  do not form a feasible solution, algorithm  $A$  has to keep the first item  $(1, 1)$  and hence the total cost is at least 1, while the optimal cost is  $1/2$  by keeping two items  $(1/4, 1/2)$  and  $(3/4, 0)$ , which is again a contradiction.

In general, after algorithm  $A$  accepts items  $(1, 1)$  and  $(1/2^2, 1/2), (1/2^3, 1/2^2), \dots, (1/2^k, 1/2^{k-1})$ , the next item  $(1/2^{k+1}, 1/2^k)$  is given. Then by applying a similar argument, algorithm  $A$  has to keep all the items given so far.

We finally prove that the total cost in knapsack will exceed  $r$  after some rounds, while the optimal cost is 1, by keeping only item  $(1, 1)$ . Let  $k$  be an integer with  $2 - 2^{-k} > r$ . Then after  $(k + 1)$ -round, the total cost by algorithm  $A$  is

$$1 + 1/2 + 1/4 + \dots + 2^{-k} = 2 - 2^{-k} > r.$$

This contradicts the assumption. □

## 5 A special case where the cost equals the size

In this section, we focus on the case where every item has its cost equal to its size. We first give a lower bound 1.618 and then propose an online algorithm which matches the lower bound.

**Lemma 10** *If any item has its cost equal to its size, then no deterministic algorithm for the online Min-Knapsack problem has competitive ratio  $r < 1 + q$  ( $\approx 1.618$ ), where  $q$  is the golden ratio, i.e.,  $q$  is the positive root for  $q^2 + q = 1$*

*Proof.* Assume an online algorithm  $A$  has its competitive ratio  $r < 1 + q$ . The first item with size  $1 + q$  is given, by Lemma 1, algorithm  $A$  has to accept the item. Let  $\epsilon = 1 + q - r$ . Then the second item with size  $1 - \epsilon$  is given. If algorithm  $A$  accepts the second item, then the total cost by algorithm  $A$  is  $2 + q - \epsilon$  and its competitive ratio is  $(2 + q - \epsilon)/(1 + q) > 1 + q - \epsilon$ .

On the other hand, if algorithm  $A$  rejects the second item, then the adversary will give the third item with size  $\epsilon$  and stop the input sequence. In this case, we can see that the cost by algorithm  $A$  is at least  $1 + q$ , while the optimal cost is 1 by keeping two items with size  $1 - \epsilon$  and  $\epsilon$ .

In either case, algorithm  $A$  has competitive ratio larger than  $1 + q - \epsilon (= r)$ , which contradicts the assumption.  $\square$

Let us then construct an online algorithm. Note that any optimal cost is at least 1, since any item has its cost equal to its size. Thus to design an online algorithm with the competitive ratio 1.618, we exploit the following strategies:

- i) if there is a feasible solution with the cost at most 1.618, then keep it and reject all the others (including the forecoming items);
- ii) Otherwise (i.e., if any feasible solution has the cost larger than 1.618), then keep a feasible solution such that the cost is not too far from the optimal cost.

An item  $d$  is called *x-large*, *large*, *medium*, and *small* if  $s(d) > 1 + q$ ,  $1 \leq s(d) \leq 1 + q$ ,  $q < s(d) < 1$ , and  $0 < s(d) \leq q$ , respectively. Let us denote by  $XL$ ,  $L$ ,  $M$ ,  $S$  the set of x-large, large, medium and small items, respectively. In other words,

$$\begin{aligned} XL &= \{d \in D \mid s(d) > 1 + q\}, & L &= \{d \in D \mid 1 \leq s(d) \leq 1 + q\}, \\ M &= \{d \in D \mid q < s(d) < 1\}, & S &= \{d \in D \mid s(d) \leq q\}. \end{aligned}$$

Similarly to the previous sections, let  $D(t) = \{d_1, \dots, d_t\}$  and let  $F(t)$  denote the set of items in the knapsack after the  $t$ -th round. Let  $t_0$  be the first time when  $D(t)$  has a feasible solution.

By Lemma 1, our algorithm accepts all the items before  $t_0$ , i.e.,  $F(t) = D(t)$ . At time  $t (\geq t_0)$ , our algorithm keeps at most two medium items and at most one x-large item, i.e.,  $|F(t) \cap M| \leq 2$  and  $|F(t) \cap XL| \leq 1$ . If two medium items are contained in the knapsack, no x-large item is kept in the knapsack, i.e., if  $|F(t) \cap M| = 2$  then  $F(t) \cap XL = \emptyset$ . Moreover, once we find a feasible solution  $U$  with the cost within  $[1, 1 + q]$ , then our algorithm only keeps this feasible solution in the knapsack and reject all the forthcoming items, i.e.,  $F(t') = U$  for  $t' \geq t$ . For example, if  $d_t$  is large and  $c(F(t-1)) \notin [1, q]$ , then  $F(t') = \{d_t\}$  for  $t' \geq t$ . Our algorithm always accepts the small items before finding a feasible solution with the cost within  $[1, 1 + q]$ . The following table shows three possible patterns for the number of x-large, large, medium and small items in the knapsack.

Table 2: Three possible patterns for the number of x-large, large, medium and small items in the knapsack

pattern	small	medium	large	x-large
1	0	0	1	0
2	$\geq 0$	2	0	0
3	$\geq 0$	$\leq 1$	0	$\leq 1$

Let us now describe our algorithm.

Algorithm C for  $t \geq t_0$

1. If  $1 \leq c(F(t-1)) \leq 1 + q$ , then  $F(t) := F(t-1)$  and halt.
2. If  $d_t \in XL$ , /\* we have three cases \*/
  - (a) If  $s(F(t-1)) < 1$ , then  $F(t) := F(t-1) \cup \{d_t\}$ .
  - (b) If  $|F(t-1) \cap M| = 2$ , then  $F(t) := F(t-1)$ .
  - (c) If  $F(t-1) \cap XL = \{e\}$ , then construct  $F(t)$  from  $F(t-1) \cup \{d_t\}$  by removing the largest  $x$ -large item  $f$  (i.e.,  $f = d_t$  if  $s(d_t) \geq s(e)$ ; otherwise,  $f = e$ )
3. If  $d_t \in L$ , then  $F(t) := \{d_t\}$  and halt. /\* we have only one case \*/
4. If  $d_t \in M$ , /\* we have four cases \*/
  - (a) if  $s(d_t) + s(F(t-1) \cap S) \geq 1$  then let  $F(t)$  be a feasible solution with cost at most  $1 + q$ .
  - (b) if  $|F(t-1) \cap M| = 2$ , then construct  $F(t)$  from  $F(t-1) \cup \{d_t\}$  by removing the smallest medium item  $f$ .
  - (c) if  $|F(t-1) \cap M| = 1$ , then  $F(t) := (F(t-1) \cup \{d_t\}) \setminus XL$ .
  - (d) if  $F(t-1) \cap M = \emptyset$ , then  $F(t) := F(t-1) \cup \{d_t\}$ .
5. If  $d_t \in S$ , /\* we have three cases \*/
  - (a) If  $F(t-1) \cap M = \emptyset$  and  $s(F(t-1) \cap S) + s(d_t) \geq 1$ , then let  $F(t)$  be a feasible solution with cost at most  $1 + q$ .
  - (b) If  $F(t-1) \cap M \neq \emptyset$  and  $s(e) + s(F(t-1) \cap S) + s(d_t) \geq 1$  for some medium  $e \in F(t-1)$ , then let  $F(t)$  be a feasible solution with cost at most  $1 + q$ .
  - (c) Otherwise,  $F(t) := F(t-1) \cup \{d_t\}$ .

We first prove several properties of the online algorithm to analyze the competitive ratio of the algorithm.

**Lemma 11** *For  $U \subseteq D$ , if  $s(U \cap S) \geq 1$  or  $s(U \cap S) + s(e) \geq 1$  for some  $u \in U$ , then  $U$  contains a feasible solution with the cost at most  $1 + q$ .*

*Proof.* Let us consider the case  $s(U \cap S) \geq 1$  only, since the other case can be treated similarly.

Let  $U' = U \cap S$ . If  $s(U') > 1 + q$ , then a feasible solution can be constructed by repeatedly removing an item from  $U'$  until the cost is at most  $1 + q$ . It is always possible since a small item has size at most  $q$ .  $\square$

The above lemma ensures that Steps 4a, 5a, and 5b are always possible. By the same reason, Step 2 has only three cases.

**Lemma 12** *At any time  $t \geq t_0$ , we have  $s(F(t)) \geq 1$ .*

*Proof.* We prove this lemma by induction on  $t$ . At time  $t = t_0$ ,  $s(F(t-1)) < 1$  and hence the execution of the algorithm for time  $t$  passes through either Step 2a, 3, 4a, 4c, 4d or 5. It is not difficult to see that  $s(F(t)) \geq 1$  when the execution passes through Step 2a, 3, 4a, 4d, or 5. Finally, if Step 4c is passed, then we have  $s(F(t)) \geq 1$ , since  $F(t-1)$  has no x-large item.

Assume the theorem holds for any time  $t_1 \geq t_0$ , i.e.,  $s(F(t_1)) \geq 1$ . Consider the next time  $t = t_1 + 1$ . It is not difficult to see that  $s(F(t)) \geq 1$  holds if the execution of the algorithm for  $t$  passes through Steps 1, 2, 3, 4a, 4d, or 5. If the execution passes through Step 4b or 4c, then  $|F(t) \cap M| \geq 2$  and hence  $s(F(t)) \geq 1$ .  $\square$

**Lemma 13** *At any time  $t \geq t_0$ , algorithm  $C$  satisfies the following three conditions.*

- (i)  $|F(t) \cap M| \leq 2$ , and furthermore if  $F(t) \cap M \neq \emptyset$ , then  $F(t)$  contains the largest item in  $D(t) \cap M$ .
- (ii)  $|F(t) \cap XL| \leq 1$ , and furthermore if  $F(t) \cap XL \neq \emptyset$ , then  $F(t)$  contains the smallest item in  $D(t) \cap XL$ .
- (iii) If  $|F(t) \cap M| = 2$ , then  $F(t) \cap XL = \emptyset$  for any  $t' \geq t$ .

Since it is not difficult to have the above claims, we skip the proof.

**Lemma 14** *Let  $t$  be an integer with  $t \geq t_0$ . if the execution of algorithm  $C$  for  $t+1$  does not pass through Step 1, then we have*

- (i)  $F(t) \cap L = \emptyset$  holds.
- (ii)  $s(f) + s(F(t) \cap S) < 1$  holds for any medium item  $f$  in  $F(t)$ .
- (iii)  $s(F(t) \cap S) < 1$  holds, and furthermore we have  $F(t) \cap S = D(t) \cap S$ .
- (iv) If  $F(t) \cap XL \neq \emptyset$ , then  $F(t) \cap M = D(t) \cap M$  with  $|F(t) \cap M| \leq 1$ .
- (v) If  $|F(t) \cap M| = 2$ , then  $s(F(t)) < 2$  and  $OPT(D(t)) \geq 2q$ .

*Proof.* It is not difficult to see Claims (i)  $\sim$  (iv), we only prove Claim (v).

If  $|F(t) \cap M| = 2$ , then  $F(t) \cap XL = \emptyset$  by Lemma 13 (iii). From (i) and (ii) in the lemma, we have  $s(F(t)) < 1 + 1 = 2$ . Let us finally show  $OPT(D(t)) \geq 2q$ . By (i) in the lemma, no optimal solution contains a large item. If an optimal solution contains an x-large item, then we have  $OPT(D(t)) \geq 1 + q > 2q$ . On the other hand, if an optimal solution contains two medium items, then we have  $OPT(D(t)) \geq 2q$ . Thus if  $OPT(D(t)) < 2q$ , then any optimal solution consists of at most 1 medium item and small items. However, by Lemma (ii) and (iii) in the lemma, this is impossible.  $\square$

**Theorem 5** *Algorithm  $C$  has competitive ratio 1.618, which matches the lower bound.*

*Proof.* Let  $R_t$  be the competitive ratio after the  $t$ th round. We prove that  $R_t \leq 1 + q$  for any time  $t \geq t_0$ .

By Lemma 12, there is always a feasible solution in the knapsack  $F(t)$ , i.e.,  $R_t < +\infty$ . If the execution of the algorithm passes through Step 3, 4a, 5a or 5b, then we have  $R_t \leq 1.618$  by Lemma 11.

Let us consider the remaining cases: 2a, 2b, 2c, 4b, 4c, 4d and 5c.

Case 2a (i.e., the execution passes through Step 2a). Then  $t = t_0$  holds by Lemma 12. This implies

$$s(F(t_0))/OPT(D(t_0)) < (1 + s(d_{t_0}))/s(d_{t_0}) < 1 + q.$$

Cases 2b, 4b, and 4c. In either case,  $F(t)$  contains two medium items. Thus by Lemma 14 (v), we have

$$s(F(t))/OPT(D(t)) < 2/2q = 1 + q.$$

Case 2c. Then  $F(t)$  contains an x-large item. By Lemma 14 (i), (iii) and (iv), it holds that  $F(t) \cap (S \cup M \cup L) = D(t) \cap (S \cup M \cup L)$ ,  $F(t) \cap L = \emptyset$  and  $|F(t) \cap M| \leq 1$ . Then by Lemma 14 (ii), we have  $s(D(t) \cap (S \cup M)) = s(F(t) \cap (S \cup M \cup L)) < 1$ , which implies that an optimal solution consists of a smallest x-large item only. Let  $x$  be such an item. Then by Lemma 13 (ii),  $x \in F(t)$ , and hence we have

$$s(F(t)) = s(x) + s(F(t) \cap (S \cup M \cup L)) < OPT(D(t)) + 1 \leq (1 + q)OPT(D(t)),$$

where the last inequality follows from  $OPT(D(t)) \geq 1 + q$ .

Case 4d. Note that  $F(t)$  contains an x-large item, since otherwise we have  $s(F(t)) < 1$ , a contradiction with Lemma 12. Similarly to Case 1c, we can show that the competitive ratio is at most  $1 + q$ .

Case 5c. Since  $s(F(t)) \geq 1$ ,  $F(t)$  must contain two medium items or one x-large item. Thus by applying a similar argument above, we can bound competitive ratio by  $1 + q$ .  $\square$

**Acknowledgments** The first author wishes to thank Chao Peng for his useful comments on improving the presentation of the paper.

## References

- [1] L.G. Babat, Linear functions on the N-dimensional unit cube, Dokl. AKad. Nauk SSSR 222, pp.761-762, 1975. (Russian)
- [2] J. Csirik, J.B.G. Frenk, M. Labbé, S. Zhang, Heuristics for the 0-1 Min-Knapsack problem, Acta Cybernetica, 10(1-2):15-20, 1991.
- [3] M.M. Güntzer, D. Jungnickel, Approximate minimization algorithms for the 0/1 knapsack and subset-sum problem, Operations Research Letters, 26:55-66, 2000.
- [4] G. Gene, E. Levner, Complexity of approximation algorithms for combinatorial problems: a survey, ACM SIGACT News Volume 12, Issue 3:52-65, 1980.
- [5] O.H. Ibarra and C.E. Kim, Fast approximation algorithms for the knapsack and sum of subset problems, Journal of the ACM, 22:463-468,1975.

- [6] K. Iwama and S. Taketomi, Removable online knapsack problems, Proc. ICALP2002, LNCS 2380, pp.293-305, 2002.
- [7] K. Iwama, G. Zhang, Optimal resource augmentations for online knapsack, APPROX-RANDOM 2007, pp.180-188 (2007).
- [8] H. Kellerer, U. Pferschy, D. Pisinger, Knapsack Problems, Springer, 2004.
- [9] M.-Y. Kao, J.H. Reif and S.R. Tate, Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem, Information and Computation, 131(1), pp 63-80, 1996. (Preliminary version appeared in SODA 1993)
- [10] G.S. Lueker, Average-case analysis of off-line and on-line knapsack problems, Proc. Sixth Annual ACM-SIAM SODA, pp.179-188, 1995.
- [11] A. Marchetti-Spaccamela and C. Vercellis, Stochastic on-line knapsack problems, Math. Programming, Vol. 68 (1, Ser. A), pp.73-104, 1995.
- [12] R. Motwani, P. Raghavan, Randomized algorithms, chap. 13, (online algorithms), Cambridge, 2005.
- [13] J. Noga, V. Sarbua, An online partially fractional knapsack problem, ISPAN 2005, pp. 108-112, 2005.