Online Minimization Knapsack Problem

Xin Han and 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

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 and memoryless algorithm for the problem, which contrasts to the result for the online maximization knapsack problem that no online algorithm has a bounded competitive ratio [8].
(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 [9]. 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 [9]. Among them, the *minimization* knapsack problem (Min-Knapsack) is one of the most natural ones (see [1,2,3,4] and [9, 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.

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

E. Bampis and K. Jansen (Eds.): WAOA 2009, LNCS 5893, pp. 182–193, 2010.

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) [1,4,9]. As for the online maximization knapsack problem, it was first studied on average case analysis by Marchetti-Spaccamela and Vercellis [12]. 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 [11] further improved the expected difference to $O(\log n)$ under a fairly general condition on the distribution. Recently, Iwama and Taketomi [7] 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 [8] 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 and Multi Knapsacks were also investigated [8,14,5].

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 Δ 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 contrasts with the result for the online Max-Knapsack that no online algorithm has a bounded competitive ratio [8], 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 improve the algorithm,

it has to keep items with the low total cost. However, this makes it difficult to guess the optimal value, since the item removed cannot be reused, even for guessing the optimal value. We devise the following strategy to overcome this difficulty. At each time, i) we guess the optimal value within $O(1)$ factor by repeatedly solving fractional Max-Knapsack problems to maximize the total size subject to bounded costs with respect to the items in the knapsack, together with the coming item, and ii) in order to find items to be kept, for each $j \geq 0$ we construct a subset F_j of items by solving the fractional Max-Knapsack problem subject to 2^{2-j} times the optimal cost, we keep items in $\bigcup_{j\geq 0} F_j$. We guarantee that each class F_j has cost at most 2^{2-j} times, which implies that the total cost in the knapsack is at most 8 times the optimal cost. Since the knapsack always contains a feasible solution of the Min- Knapsack problem, the procedure above leads to 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 ≈ 5.44. We finally consider the case in which each item has its cost equal to its size. Similarly to the online Max-Knapsack problem [7], we show that the online Min-Knapsack problem admits 1.618-competitive deterministic 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

<table>
<tr><td colspan="3" rowspan="2"></td><td colspan="2">Max-Knapsack</td><td colspan="2">Min-Knapsack</td></tr>
<tr><td>lower bound</td><td>upper bound</td><td>lower bound</td><td>upper bound</td></tr>
<tr><td colspan="3">offline</td><td colspan="2">FPTAS [6]</td><td colspan="2">FPTAS [1]</td></tr>
<tr><td rowspan="4">online</td><td rowspan="2">non-removable</td><td>general</td><td colspan="2">unbounded [7]</td><td colspan="2">unbounded</td></tr>
<tr><td>size =cost</td><td colspan="2">unbounded [7]</td><td colspan="2">unbounded</td></tr>
<tr><td rowspan="2">removable</td><td>general</td><td colspan="2">unbounded [8]</td><td>2</td><td>8
2e (randomized)</td></tr>
<tr><td>size = cost</td><td>1.618 [7]</td><td>1.618 [7]</td><td>1.618</td><td>1.618</td></tr>
</table>

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.

Due to space constraints, some proofs are omitted, which can be found in the full version.

2 Preliminaries

In this section, we 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 $D = \{d_1, \ldots, d_n\}$ associated with cost $c : D \to \mathbb{R}_+$ and size $s : D \to \mathbb{R}_+$.

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, i.e, we focus on the memoryless online algorithm. Note that this assumption is strict in the sense that most online algorithms can use the items discarded for the calculations. However we adopt this setting to tackle huge input data. 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. We leave the proof in the full version.

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) *While 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.*

From Lemma 1, in the subsequent 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 Δ 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 non-negative integer t, let $D(t)$ denote the set of the first t items, i.e., $D(t) = \{d_1, \ldots d_t\}$, 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

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

$D_j(t) = D(t) \cap S_j$ 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
1. $E(t) := F(t-1) \cup \{d_t\}$.
2. Guess: Compute a value $\alpha(t)$ by an approximation algorithm (e.g., [1,2]) with $E(t)$ as the input. Set $\beta(t) := \min\{\beta(t-1), \alpha(t)\}$.
3. For each j, $F_j(t) := E(t) \cap S_j$ and if $c(F_j(t)) > 3\beta(t)$ then repeatedly remove an item with the highest cost until $c(F_j(t)) \leq 3\beta(t)$.
4. $F(t) := \bigcup_j F_j(t)$

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, where the proofs for Lemmas 2, 3 and 4 are given in the full version.

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 $q \in F_j(t)$.*

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 $F_j^*(t) \not\subseteq F_j(t)$.*

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

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 $A(D(t)) \leq O(\log \Delta)OPT(D(t))$. □

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 $D = D^{(0)}, D^{(1)}, \ldots, D^{(k)}$, 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. □

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. Note that here the definition of $F_j(t)$ is *different* from the one in the subsection 3.1.

1 A guessed value $\beta(t)$ satisfies $\beta(t) \leq r \cdot OPT(D(t))$ for some constant $r > 1$.
2 For each $j \geq 0$, $c(F_j(t)) \leq 2\beta(t)/r^j$.
3 $F(t) := \bigcup_j F_j(t)$ 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 $F(t-1)$ denote 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 ℓ be the maximum number j such that $s(F_j(t)) \geq 1$. Then we have $OPT(D(t)) \leq c(F_l(t)) \leq 2\beta(t)/r^\ell$. 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 discarded 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 $F_j(t)$.

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

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 [9] 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 $O\big(\log \frac{t \max\{c(d) | d \in D(t)\}}{\min\{c(d) | d \in D(t)\}}\big)$.

Lemma 7. *At any time $t \geq t_0$, the index ℓ 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 $t = t_1 + 1$. □

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

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

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 $FKP\Big(F_j(t), \frac{\alpha(t)}{r^j}\Big)$, by the greedy construction of $F_j(t)$ in Step 2. Therefore, we have

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

This implies

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

□

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

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

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 ℓ at Step 3. □

Theorem 2. *When* $r = 2$, *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

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

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 [10] can foil the worst case. Namely, let ξ 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^{ξ}. Let us call this randomized algorithm RB.

We shall below show that $\mathrm{E}[RB(D)]/OPT(D) \leq 2e$ for all D against an oblivious adversary[13].

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

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. We leave the details of the proof in the full version .

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

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. The proof of Lemma 10 will be given in the full version.

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 $r < 1+\mathrm{q}\,(\approx 1.618)$, where* q *is the golden ratio, i.e.,* q *is the positive root for* $\mathrm{q}^2 + \mathrm{q} = 1$*.*

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.

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

$$XL = \{d \in D \mid s(d) > 1 + \mathrm{q}\},\ L = \{d \in D \mid 1 \le s(d) \le 1 + \mathrm{q}\},$$
$$M = \{d \in D \mid \mathrm{q} < s(d) < 1\},\ S = \{d \in D \mid s(d) \le \mathrm{q}\}.$$

Similarly to the previous sections, let $D(t) = \{d_1, \ldots, d_t\}$ 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 $t\,(\ge t_0)$, our algorithm keeps at most two medium items and at most one x-large item, i.e., $|F(t) \cap M| \le 2$ and $|F(t) \cap XL| \le 1$. 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 $[1, 1+\mathrm{q}]$, then our algorithm only keeps this feasible solution in the knapsack and rejects all the forthcoming items, i.e., $F(t') = U$ for $t' \ge t$. For example, if d_t is large and $c(F(t-1)) \notin [1, 1+\mathrm{q}]$, then $F(t') = \{d_t\}$ for $t' \ge t$. Our algorithm always accepts the small items before finding a feasible solution with the cost within $[1, 1+\mathrm{q}]$. Table 2 shows three possible patterns for the number of x-large, large, medium and small items in the knapsack.

Let us now describe our algorithm.

Algorithm C for $t \ge t_0$

1. If $1 \le c(F(t-1)) \le 1 + \mathrm{q}$, 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 $s(d_t) \ge s(e)$; 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 $s(d_t) + s(F(t-1) \cap S) \ge 1$ then let $F(t)$ be a feasible solution with cost at most $1 + \mathrm{q}$.
   (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 $s(F(t-1) \cap S) + s(d_t) \ge 1$, then let $F(t)$ be a feasible solution with cost at most $1 + \mathrm{q}$.
   (b) If $F(t-1) \cap M \ne \emptyset$ and $s(e) + s(F(t-1) \cap S) + s(d_t) \ge 1$ for some medium $e \in F(t-1)$, then let $F(t)$ be a feasible solution with cost at most $1 + \mathrm{q}$.
   (c) Otherwise, $F(t) := F(t-1) \cup \{d_t\}$.

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	≥ 0	2	0	0
3	≥ 0	≤ 1	0	≤ 1

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 $1 + \mathrm{q}$.*

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

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

References

1. Babat, L.G.: Linear functions on the N-dimensional unit cube. Dokl. AKad. Nauk SSSR 222, 761–762 (1975) (Russian)
2. Csirik, J., Frenk, J.B.G., Labbé, M., Zhang, S.: Heuristics for the 0-1 Min-Knapsack problem. Acta Cybernetica 10(1-2), 15–20 (1991)
3. Güntzer, M.M., Jungnickel, D.: Approximate minimization algorithms for the 0/1 knapsack and subset-sum problem. Operations Research Letters 26, 55–66 (2000)
4. Gene, G., Levner, E.: Complexity of approximation algorithms for combinatorial problems: a survey. ACM SIGACT News 12(3), 52–65 (1980)
5. Horiyama, T., Iwama, K., Kawahara, J.: Finite-State Online Algorithms and Their Automated Competitive Analysis. In: Asano, T. (ed.) ISAAC 2006. LNCS, vol. 4288, pp. 71–80. Springer, Heidelberg (2006)
6. Ibarra, O.H., Kim, C.E.: Fast approximation algorithms for the knapsack and sum of subset problems. Journal of the ACM 22, 463–468 (1975)
7. Iwama, K., Taketomi, S.: Removable online knapsack problems. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 293–305. Springer, Heidelberg (2002)
8. Iwama, K., Zhang, G.: Optimal resource augmentations for online knapsack. In: APPROX-RANDOM 2007, pp. 180–188 (2007)
9. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer, Heidelberg (2004)
10. Kao, M.-Y., Reif, J.H., Tate, S.R.: Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. Information and Computation 131(1), 63–80 (1996); Preliminary version appeared in SODA (1993)
11. Lueker, G.S.: Average-case analysis of off-line and on-line knapsack problems. In: Proc. Sixth Annual ACM-SIAM SODA, pp. 179–188 (1995)
12. Marchetti-Spaccamela, A., Vercellis, C.: Stochastic on-line knapsack problems. Math. Programming 68(1, Ser. A), 73–104 (1995)
13. Motwani, R., Raghavan, P.: Randomized algorithms, ch.13 (online algorithms), Cambridge (2005)
14. Noga, J., Sarbua, V.: An online partially fractional knapsack problem. In: ISPAN 2005, pp. 108–112 (2005)