Vol.2010-AL-128 No.7
2010/1/26

Online Knapsack Problems with Limited Cuts

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

1. Introduction

The knapsack problem is one of the most classical and studied problems in combinatorial optimization and has a lot of applications in the real world[10]. The (classical) knapsack problem is given a set of items with weights and sizes, and the capacity value of a knapsack, to maximize the total weight 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[10]. Among them, the *minimization* knapsack problem (Min-Knapsack) is one of the most natural ones (see[1)–4)] and[10] [pp. 412-413]), that is given a set of items associated with weights and sizes, and the size of a knapsack, to minimize the total weight of selected items that cover the knapsack. In this paper, we study online maximization and minimization knapsack problems with limited cuts, in which i) items are given one by one over time, i.e., after a decision is made on the current item, the next one is given, ii) items are allowed to be cut at most $k\,(\geq 1)$ times, and iii) items are allowed to be removed from the knapsack (but once they are removed, they cannot be used partially again).

Related work: It is well-known that offline Max-Knapsack and Min-Knapsack are both NP-hard, and admit fully polynomial time approximation schemes (FPTASs)[1),4),7),10)]. 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. Iwama and Taketomi[8)] 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 the best possible by providing a lower bound 1.618 for this case. For the general case, Iwama and Zhang[9)] showed that no algorithm for online Max-Knapsack has a bounded competitive ratio, even if the removal condition is allowed. Recently, Han and Makino[6)] obtained an upper bound 8 and a lower bound 2 for minimization knapsack problem. Iwama and Zhang[9)] presented the competitive ratio for the online Max-Knapsack problem with resource augmentation. Noga and Sarbua[13)] studied an online partially fractional knapsack problem with resource augmentation, in which items are allowed to be cut at most once (i.e., $k = 1$), only before they are packed into the knapsack, whereas in our model items are allowed to be cut any time (but at most $k\,(\geq 1)$ times). They gave an upper bound $2/m$ and proved the bound is the best possible, where $m \geq 1$ is the capacity of the knapsack used by online algorithms while the optimal offline algorithm uses a unit capacity knapsack. The online knapsack problem of using extra a bin and allowing to exchange items between two bins was studied by Horiyama, Iwama and Kawahara[5)].

Our contributions: For the online maximization knapsack, we propose a simple greedy online algorithm, in which whenever a cut is necessary on an item, we almost cut off the fraction of size $\frac{1}{k+1}$ from the item. We show that our greedy algorithm is $\frac{k+1}{k}$-competitive, and it is the best possible by giving a lower bound of the cometitive ratio. We extend this result to to the model with

resource augmentation. In the resource augmentation model, we show that the online maximization knapsack problem is $\max\{1, \frac{k+1}{m(k+1)-1}\}$-competitive (i.e., we present a $\max\{1, \frac{k+1}{m(k+1)-1}\}$-competitive algorithm and show that it is the best possible). When $k = 1$ and $1 \leq m < 2$, the competitive ratio $\frac{2}{2m-1}$ in our model is smaller than the ratio $\frac{2}{m}$ in the partial cut model discussed in[13]. This implies that our model for $k = 1$ is more powerful than the model given in[13].

For the minimization knapsack problem, we show that no online algorithm can have a constant competitive ratio, i.e., our cut condition does not help solving the problem. This result will appear in a full version.

Table 1 summarizes the current results on online Max-Knapsack with resource augmentation, where the bold letters represent the results obtained in this paper, $m\,(\geq 1)$ denotes the capacity used by online algorithms and k denotes the number of limited cuts to be allowed.

Table 1 The current results on online Max-Knapsack with resource augmentation

Max-Knapsack	$k = 0$	$k = 1$ in the partial cut model	$k \geq 1$ in our model
Lower Bound	$1/(m-1)$[9]	$2/m$[13]	$\boldsymbol{(k+1)/(m(k+1)-1)}$
Upper Bound	$1/(m-1)$[9]	$2/m$[13]	$\boldsymbol{(k+1)/(m(k+1)-1)}$

2. Preliminaries

In this section, we formally define our problems and review the basic concepts for the online algorithms.

Problem Max-Knapsack (resp., Min-Knapsack)

Input: A set of items $L = \{a_1, \ldots, a_n\}$ associated with weight $w : L \to \mathbb{R}_+$ and size $s : L \to \mathbb{R}_+$.

Output: A set of items $F \subseteq L$ that maximizes $w(F)$ subject to $s(F) \leq 1$ (resp., that minimizes $w(F)$ subject to $s(F) \geq 1$).

Here, for a set $U \subseteq L$, let $w(U) = \sum_{u \in U} w(u)$ and $s(U) = \sum_{u \in U} s(u)$, and we assume w.l.o.g. that the size of the knapsack is 1. The fractional version of the Max-Knapsack (resp., Min-Knapsack) is given as follows: $\max \sum_{u \in L} w(u)x(u)$ s.t. $\sum_{u \in L} s(u)x(u) \leq 1$ and $0 \leq x(u) \leq 1$ $(u \in L)$ (resp., $\min \sum_{u \in L} w(u)x(u)$ s.t. $\sum_{u \in L} s(u)x(u) \geq 1$ and $0 \leq x(u) \leq 1$ $(u \in L)$).

In our 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 a_t, the next one a_{t+1} is given. Besides this, our model satisfies the *removal* and *cut* conditions.

Removal condition: The items in the knapsack are allowed to be removed, where the items removed cannot be used again.

Cut condition: The current item and the items in the knapsack are allowed to be cut, where the part of the item cut off cannot be used again, and during the whole process, each item can be cut at most $k\,(\geq 1)$ times. Here k is a given positive integer.

By the cut condition, the knapsack keeps a set of fractional items, and hence our problems can be regared as the online fractional knapsack problems, rather than online 0-1 knapsack problems.

We analyze online algorithms by using one of the standards: the competitive ratio. Given an input sequence L and an online algorithm A, for the maximization problem, the *competitive ratio* of algorithm A is defined as follows:

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

and for the minimization problem, 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 weights obtained by an optimal algorithm and the algorithm A, respectively.

3. Online Maximization Knapsack with Limited Cuts

3.1 A simple greedy algorithm A

The main ideas of our algorithm are as follows: when a new item is arrived, we apply a greedy algorithm to select items from the knapsack together with the new item. If the total size of the resulting items is greater than the capacity of the knapsack, then we cut the less efficient item, say b in the knapsack. Let $s(b)$ be the size of item b. The rule of cutting is below: if $s(b) > 1$ (the capacity of

the knapsack) then we cut a fraction from b such that the remaining size $s(b)$ is exactly $\frac{k}{k+1}$, else cut a fraction of size $\min\{\frac{1}{k+1}, s(b)\}$ from item b. Then we repeatedly cut off item b, until the total size becomes at most the capacity of the knapsack.

Let $L = \{a_1, a_2, ..., a_n\}$ be the online input. Assume that items $a_1, ..., a_{i-1}$ have been dealt by our algorithm. Let B_{i-1} be the set of items in the knapsack. The execution of our algorithm on item a_i is the following.

Algorithm: A

(1) If $s(B_{i-1}) \geq \frac{k}{k+1}$ and the density (weight/size) of item a_i is not larger than the smallest density of items in B_{i-1} then reject item a_i immediately, else $B_i^{'} := B_{i-1} \cup \{a_i\}$,
   (a) Rename all the items in B_i' as $b_1, b_2, ...$ such that $w(b_1)/s(b_1) \geq w(b_2)/s(b_2) \geq \cdots$, where $w(b_j)$ and $s(b_j)$ respectively denote the weight and size of fractional item b_j.
   (b) Find a smallest index x such that $\sum_{h=1}^{x} s(b_h) > 1$, remove all the items with index larger than x in $B_i^{'}$.
   (c) If $\sum_{h=1}^{x-1} s(b_h) \geq \frac{k}{k+1}$, then remove item b_x. Else repeatedly chop off b_x by the following way until the total size in B_i' becomes at most 1: if $s(b_x) \leq 1$ chop off by a fraction of size $\frac{1}{k+1}$ from item b_x else chop off by a fraction such that the remaining size of item b_x is exactly $\frac{k}{k+1}$.

(2) Update set B_i.

Theorem 1 The competitive ratio of algorithm A is $\frac{k+1}{k}$.

Proof. It is not difficult to see that if an item originally has size at most 1 we never cut the item more than k item before it is totally removed, if an item originally has size larger than 1, this is also true since after the first cutting on the item the remaining size of the item is eactly $\frac{k}{k+1}$.

So next we need to prove that for all $1 \leq i \leq n$,

$$\frac{OPT(L_i)}{A(L_i)} \leq \frac{k+1}{k},$$

where $L_i = \{a_1, a_2, ..., a_i\}$ is the input just after time i, $OPT(L_i)$ and $A(L_i)$ are the total weights by an offline optimal and our online algorithms, respectively.

Just after time i, let B_i be the set of pieces in the knapsack. And let R_i be the set of pieces which have been discarded by algorithm A. Observe that algorithm A always uses a greedy policy to select items. If both R_i and B_i are not empty, for any two pieces $q \in R_i$ and $p \in B_i$, we have

$$\frac{w(p)}{s(p)} \geq \frac{w(q)}{s(q)}, \tag{1}$$

where $w(p)$ (resp., $w(q)$) is the weight of fractional item p (resp., q) and $s(p)$ (resp., $s(q)$) is the size of fractional item p (resp., q). Moreover if R_i is not empty, we have

$$s(B_i) \geq \frac{k}{k+1}, \tag{2}$$

where $s(B_i)$ is the total size in set B_i.

If R_i is empty, then we have

$$A(L_i) = w(B_i) = OPT(L_i),$$

otherwise by (1) and (2), we immediately have

$$A(L_i) = w(B_i) \geq (1 - \frac{1}{k+1})OPT(L_i).$$

Hence this theorem holds. □

3.2 A tight lower bound for the competitive ratio of the maximum knapsack

Surprisingly, the upper bound $\frac{1+k}{k}$ by online algorithm A is the best we can do, i.e., there exists no online algorithm with a competitive ratio less than $\frac{1+k}{k}$. We prove this in this subsection.

Assume there is an online algorithm with a competitive ratio c, which is less than $\frac{1+k}{k}$. Then the main ideas are as below:

(1) We force the online algorithm to accept a large item with a low density and a unit size.

(2) Sequentially, small items follow and their densities gradually increase, after some steps the online algorithm has to cut the large item to save space for small items otherwise its competitive ratio reaches to $\frac{1+k}{k}$; and more if the large item is cut then the size of the fraction cut off has to be less than $\frac{1}{k+1}$, otherwise the competitive ratio is as least $\frac{1+k}{k}$;

(3) We keep doing the above operation k times, i.e., the online algorithm cuts

the large item k times and every time a portion of size less than $\frac{1}{k+1}$ is cut off.

(4) We finally continue to give new small items and increase their densities, then the online algorithm rejects the large item or does not, in both cases, we can prove that the online algorithm has a competitive ratio larger than c.

Theorem 2 No online algorithm has a competitive ratio smaller than $\frac{1+k}{k}$.

Proof. Assume there is an online algorithm A with a competitive ratio $c = \frac{1+k}{k+r} < \frac{1+k}{k}$, where $r > 0$. We prove that there is an input L such that $OPT(L)/A(L) > c$. The ideas to construct the list L are similar with the ones in[9].

In the input L, there are two kinds of sizes 1 and ϵ, i.e., *large* and *small*, where $\epsilon > 0$ is a sufficiently small and such that $\epsilon < \frac{r}{3(k+1)}$ and $\frac{1}{k\epsilon}$ is an integer. The input L is formed by phases. In phase 0, there is only a large item $(1, 1)$. For any $i > 0$, each phase i has $1/\epsilon$ items and each item has size ϵ and weight $\epsilon + i\epsilon^2$. Namely, the input L is below:

$$\begin{array}{c}(1,1)\\ (\epsilon+\epsilon^2,\epsilon),\ \ (\epsilon+\epsilon^2,\epsilon),...,(\epsilon+\epsilon^2,\epsilon)\\ (\epsilon+2\epsilon^2,\epsilon),\ (\epsilon+2\epsilon^2,\epsilon),...,(\epsilon+2\epsilon^2,\epsilon)\\ ...\\ (\epsilon+i\epsilon^2,\epsilon),\ (\epsilon+i\epsilon^2,\epsilon),...,(\epsilon+i\epsilon^2,\epsilon)\\ ...\end{array}$$

Note that the information of the input L is gradually known to the online algorithm A and the input can stop at any step if the online algorithm performs poorly. Moreover online algorithm A does not know the future information of L and it can only use the information known so far.

We are going to prove that if there is a cut by the online algorithm A, then the size of the portion cut off is less than $1/(k+1)$. Let $OPT(i, j)$ be the optimal value just after the j-th item of phase i is given, where $1 \le j \le \frac{1}{\epsilon}$ and $i \ge 1$. It is not difficult to see

$$OPT(i,j) = (\frac{1}{\epsilon} - j)(\epsilon + (i-1)\epsilon^2) + j(\epsilon + i\epsilon^2) = 1 + \epsilon(i-1) + j\epsilon^2. \quad (3)$$

Lemma 1 In order to achieve c-competitive, from phase 0 to phase $\frac{1}{\epsilon k}$, algorithm A has to cut the large item or discard it from the knapsack. If the algorithm A cuts the large item, then at the first cutting, the portion cut off has size smaller than $\frac{1}{k+1}$.

Lemma 2 Assume the large item has been cut $j < k$ times before phase $i_0 \ge 0$ and its remaining size in the knapsack is $x \ge \frac{1}{k+1}$. If there exists an integer $i > i_0$ such that

$$\frac{OPT(i-1, \frac{1}{\epsilon})}{x + (1-x)\frac{\epsilon + i\epsilon^2}{\epsilon}} \ge \frac{k+1}{k},$$

then algorithm A has to cut the large item or discard it from the knapsack before phase i. If the algorithm A makes its $(j+1)$-th cutting on the large item, then the size of the portion cut off is smaller than $\frac{1}{k+1}$.

(The above lemmas will be given in a full version. Here we omit the details of the proofs.)

Again, let x be the remaining size of the large item in the knapsack after the previous cutting. If $x > \frac{1}{k+1}$, there always exists an i such that the condition in Lemma 2 holds, i.e.,

$$\begin{aligned}\frac{OPT(i-1, \frac{1}{\epsilon})}{x + (1-x)\frac{\epsilon + i\epsilon^2}{\epsilon}} &\ge \frac{1 + (i-1)\epsilon}{x + (1-x)(1+i\epsilon)} \quad \text{by (3)}\\ &= \frac{1 + (i-1)\epsilon}{1 + i\epsilon - x\epsilon} \ge \frac{k+1}{k},\end{aligned}$$

where the last inequality holds directly from $i \ge \frac{1}{\epsilon} \cdot \frac{1+k\epsilon}{(k+1)x-1}$.

Then by induction, we can see that the condition in Lemma 2 always holds before the large item has been cut k times.

By Lemmas 1 and 2, every time when the algorithm A cuts the large item, it cuts a portion of size less than $1/(k+1)$. Assume that the large item is discarded at size x. Therefore $x > 1 - k \times \frac{1}{k+1} = \frac{1}{k+1}$. Once the large item is discarded at phase $i > 0$, we stop the input L. At this step, $A(L) \le (1 - x + \epsilon)(1 + i\epsilon)$ and $OPT(L) \ge 1 + (i-1)\epsilon$. Then the competitive ratio of algorithm A is at least

$$\begin{aligned}\frac{1 + \epsilon(i-1)}{(1-x+\epsilon)(1+i\epsilon)} &\ge \frac{1}{(1-x+\epsilon)(1+\epsilon)} \ge \frac{1}{(1+3\epsilon) - x(1+\epsilon)}\\ &> \frac{k+1}{k + 3\epsilon(k+1)} > \frac{k+1}{k+r} = c,\end{aligned}$$

where the second inequality holds from $(1+\epsilon)^2 \leq 1+3\epsilon$ and the third one holds from $x > \frac{1}{k+1}$ and $\epsilon > 0$.

After k times cutting, if the large item keeps staying in the knapsack after phase $i > \frac{1}{\epsilon}(\frac{1}{r}-1)$, we stop the input L just after phase i. Then by (3) $OPT(L) > \frac{1}{r}$ and $A(L)$ is at most $(1-x)OPT(L)+x$. Therefore, the competitive ratio after phase i is at least

$$\frac{OPT(L)}{(1-x)OPT(L)+x} \geq \frac{1}{(1-x)+x/OPT(L)} \geq \frac{1}{1-x(1-1/OPT(L))}$$
$$> \frac{1}{1-\frac{1-1/OPT(L)}{k+1}} = \frac{1+k}{k+1-(1-1/OPT(L))} > \frac{k+1}{k+r} = c,$$

where the third inequality follows from $x > \frac{1}{k+1}$ and $OPT(L) > 1$.

Hence, there exists an input L such that $OPT(L)/A(L) > c$, i.e., there is not an online algorithm with the competitive ratio strictly smaller than $\frac{k+1}{k}$. □

Remarks: In the model[13], the "cutting" is only allowed before packing, namely, when an item has been packed, it is not allow to cut it. In our model, there is not this restriction and we are allowed to cut items any time. So, our model is a generalization of the model in[13]. When $k = 1$, our upper and lower bounds are the same as the results in[13].

4. Resource Augmentation for the Online Maximization Knapsack with Limited Cuts

In this section, we study resource augmentation for the online maximization knapsack with limited cuts, in which the online algorithm uses a knapsack with capacity $m \geq 1$, while the offline algorithm uses a knapsack with capacity 1. We provide the competitive ratio in this model.

4.1 A simple greedy algorithm

Let $L = \{a_1, a_2, ..., a_n\}$ be the online input. Assume that items $a_1, ..., a_{i-1}$ have been dealt by our algorithm. Let B_{i-1} be the set of items in the knapsack. The execution of our algorithm on item a_i is the following.

Algorithm: B for Resource Augmentation

(1) If $s(B_{i-1}) \geq m - \frac{1}{k+1}$ and the density (weight/size) of item a_i is not larger than the smallest density of items in B_{i-1} then reject item a_i immediately, else $B_i^{'} := B_{i-1} \cup \{a_i\}$,
   (a) Rename all the items in $B_i^{'}$ as $b_1, b_2, ...$ such that $w(b_1)/s(b_1) \geq w(b_2)/s(b_2) \geq \cdots$, where $w(b_j)(s(b_j))$ is the weight(size) of item b_j.
   (b) Find a smallest index x such that $\sum_{h=1}^{x} s(b_h) > m$, remove all the items with index larger than x in $B_i^{'}$.
   (c) If $\sum_{h=1}^{x-1} s(b_h) \geq m - \frac{1}{k+1}$, then remove item b_x. Else repeatedly chop off b_x by the following way until the total size in $B_i^{'}$ becomes at most m: if $s(b_x) \leq 1$ chop off by a fraction of size $\frac{1}{k+1}$ from item b_x else chop off by a fraction such that the remaining size of item b_x is exactly $\frac{k}{k+1}$.
(2) Update set B_i.

Observe that if there are some pieces of items discarded, then the total size in the knapsack is at least $m - \frac{1}{k+1}$. Due to the greedy police used in the above algorithm, we have the density of any item in the knapsack is not lower than the density of any item discarded. Then by the similar approach with Theorem 1, we have the following theorem.

Theorem 3 The competitive ratio of algorithm B is $\max\{1, \frac{k+1}{m(k+1)-1}\}$.

Remarks: for $k = 1$, our upper bound $\frac{2}{2m-1}$ is better than the bound $\frac{2}{m}$ in[13]. Note that the bound $\frac{2}{m}$ is tight for the model of only allowing cutting items before they have been packed into the knapsack[13]. Our result also implies that our model of allowing cutting items any time is more powerful than the model in[13].

4.2 A tight lower bound

In this subsection, we prove that the ratio $\max\{1, \frac{k+1}{m(k+1)-1}\}$ is the best possible ratio we can do, i.e., there is not an online algorithm with a competitive ratio strictly less than this ratio. The main ideas are the same with the model without resource augmentation. Here we only consider the non-trivial case $1 \leq m < \frac{k+2}{k+1}$ here. The details of the proof are in Appendix.

Theorem 4 No online algorithm has a competitive ratio smaller than

$\frac{k+1}{m(k+1)-1}$.

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 and S. Zhang, Heuristics for the 0-1 Min-Knapsack problem, *Acta Cybernetica*, 10(1-2):15-20, 1991.
3) M. M. Güntzer and D. Jungnickel, Approximate minimization algorithms for the 0/1 knapsack and subset-sum problem, *Operations Research Letters*, 26:55-66, 2000.
4) G. Gene and E. Levner, Complexity of approximation algorithms for combinatorial problems: a survey, *ACM SIGACT News* Volume 12, Issue 3:52-65, 1980.
5) T. Horiyama, K. Iwama and J. Kawahara, Finite-State Online Algorithms and Their Automated Competitive Analysis, *ISAAC*, LNCS 4288, pp. 71-80, 2006.
6) X. Han and K. Makino, Online minimization knapsack problem, Mathematical Engineering Technical Reports, Tokyo Univeristy, *METR* 2009-04, to appear in *WAOA* 2009.
7) 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.
8) K. Iwama and S. Taketomi, Removable online knapsack problems, *Proc. ICALP 2002*, LNCS 2380, pp.293-305.
9) K. Iwama and G. Zhang, Optimal resource augmentations for online knapsack, *APPROX-RANDOM* 2007, pp.180-188.
10) H. Kellerer, U. Pferschy and D. Pisinger, *Knapsack Problems*, Springer, 2004.
11) G. S. Lueker, Average-case analysis of off-line and on-line knapsack problems, *Proc. Sixth Annual ACM-SIAM SODA*, pp.179-188, 1995.
12) A. Marchetti-Spaccamela and C. Vercellis, Stochastic on-line knapsack problems, *Math. Programming*, Vol. 68 (1, Ser. A), pp.73-104, 1995.
13) J. Noga and V. Sarbua, An online partially fractional knapsack problem, *ISPAN* 2005, pp. 108-112.