Online Knapsack Problems

Deeparnab Chakrabarty
Dept. of Comb. and Opt.
University of Waterloo
Waterloo, ON N2L 3G1, Canada
deepc@math.uwaterloo.ca

Yunhong Zhou
Rocket Fuel Inc.
One Lagoon Dr.
Redwood Shores, CA 94065
yzhou@rocketfuelinc.com

Rajan Lukose
HP Research Labs
1501 Page Mill Rd
Palo Alto, CA 95304
rajan.lukose@hp.com

Abstract

Since no competitive online algorithms exist for general knapsack problems, we consider *online* variants of *knapsack problems* with two assumptions: (i) the weight of each item is very small compared to the knapsack capacity; (ii) the value-to-weight ratio of each item is lower and upper bounded by two positive constants L and U. We design a deterministic threshold-based algorithm for the online knapsack problem achieving a *provably optimal* competitive ratio $\ln(U/L)+1$. We also extend the online algorithm to variations of knapsack problems, include the *multiple knapsack problem*, the *multiple-choice knapsack problem*, and the *generalized assignment problem*, and discuss about their applications to sponsored search auctions[1].

1 Introduction

In this paper we consider the *online version* of variants of knapsack problems. The most basic problem we consider is the classic *0/1 knapsack problem*: Given a knapsack of capacity B, m items where item j has value v_j and weight w_j, the goal is to obtain a subset of items having maximum value with the total weight being at most B. The most general problem we consider is the *generalized assignment problem* (GAP): Given a set A of n knapsacks (or bins) of capacity $B_1, \cdots, B_n$, a set Q of m items with item j having a value v_{ij} and weight w_{ij} for bin i, the goal is to find an allocation of items to bins such that the total weight in each bin does not exceed the capacity and the total value across bins is maximized.

We devise algorithms for online versions of these knapsack problems. That is, the set of items Q is not known at the beginning and items arrive one at a time. At each instant, the algorithm must decide what to do with the item (place it in a bin or discard it), and the decision once made is irrevocable - discarded items cannot be recalled back and items once allocated cannot be de-allocated.

We analyze the performance of our algorithms using *competitive analysis*. That is, given any set of items J, we compare the value obtained by our algorithms to the best value that could have been obtained by *any* (computationally all-powerful) "offline" algorithm who has complete knowledge about the set of items. Comparing with such an all powerful benchmark has its difficulties - as we make clear later, if one makes no assumption then no algorithm with constant competitive ratio is possible. We make the following two assumptions throughout the paper:

1. The weight of each item is much smaller than the capacity of the bins. In particular, there is an $\epsilon > 0$ very close to 0, such that $w_{ij}/B_i \le \epsilon$ for all $i \in A, j \in Q$. [2]

[1]A preliminary version of this work bases on this application appeared in [3]

[2]In fact, for the sake of brevity of notation and simplicity of expressions, throughout the paper all our results will be states assuming $\epsilon = 0$; all factors need to be multiplied a factor of $\texttt{error}(\epsilon)$ for the precise factor, where $\texttt{error}(\epsilon) \to 1$ as $\epsilon \to 0$. We show this factor once and avoid repeating the same argument all the time.

2. The value-to-weight ratios are neither too high, nor too small. That is, there exists $U, L > 0$ such that
$$L \leq \frac{v_{ij}}{w_{ij}} \leq U, \quad \forall\ i \in A,\ j \in Q.$$

Our main results are a $(\ln(U/L)+2)$-competitive algorithm for online generalized assignment problem (ON-GAP) and the online multiple-choice knapsack problem (ON-MCKP), and a $(\ln(U/L)+1)$-competitive algorithm for the 0/1 knapsack problem (ON-KP) and the multiple knapsack problem (ON-MKP). For the cases of the single knapsack problem, we show no algorithm, even allowing randomization, can achieve a competitive ratio better than $(\ln(U/L)+1)$. Therefore, the competitive ratios of our algorithms for ON-KP and ON-MKP are optimal, while for ON-GAP and ON-MCKP they are off from the optimal by at most 1. All our algorithms are simple and deterministic.

1.1 Related Work

The offline knapsack problem and its variations are well studied in the computer science and operations research communities. For a acomprehensive treatment of the topic, see the textbook by Kellerer et al. [6]. The classic 0/1 knapsack problem is NP-complete, as is true for most of its variants. Both the 0/1 knapsack problem (KP) and the multiple-choice knapsack problem (MCKP) accept an FPTAS [6]. The multiple knapsack problem (MKP) is slightly harder as it has only a PTAS [4] and does not have an FPTAS unless P=NP. The generalized assignment problem (GAP) is APX-hard.

The online knapsack problem was first studied by Marchetti-Spaccamela and Vercellis [9] who show that no non-trivial competitive algorithms existed for the general case of the online knapsack problem. To see this, consider the case where the capacity $B = 1$ and consider two input sequences both having two items, $\sigma_1 = \{(1,1),(0,1)\}$ and $\sigma_2 = \{(1,1),(\infty,1)\}$. Its easy to see any deterministic online algorithm will be infinitely worse than the omniscient algorithm on *at least one* of the inputs. It is not too hard to generalize this argument for randomized online algorithms as well.

Besides giving the impossibility result, [9] study the online knapsack problem in the average case setting with the values and weights of the items being picked from a fixed distribution and give an online algorithm which, in our setting, gives a value an additive factor away from the optimal. Lueker [8] improves the additive factor and gives an optimal algorithm in this setting. More general stochastic knapsack problems are considered by Papastavrou et.al [11, 7] and Van-Slyke and Young [12]. Other variants of the online knapsack problem like the removable online knapsack problems [5] and online partially fractional knapsack problems [10] have also been studied recently.

As per our knowledge, we do not know of any work on the online knapsack problem with the assumptions that we make. However, recently in a series of works, Buchbinder and Naor [1, 2] design online algorithms for fractional versions of general packing problems; our allocation problems are also packing problems. One can derive $O(\ln(U/L))$-competitive online algorithms for our problems using their framework. Nevertheless, we believe our algorithms are much simpler for the special case of knapsack problems and furthermore our results are either optimal or off from the optimal by at most an additive factor of 1.

2 The Online Knapsack Problem

In this section we first present a deterministic algorithm for the online knapsack problem achieving competitive raio of $\ln(U/L)+1$. The algorithm works against all adversaries with adaptive input sequences. Next we prove that any algorithm (either deterministic or randomized) can not achieve a competitive ratio lower than $\ln(U/L)+1$. Thus our algorithm achieves the *optimal* competitive ratio. In the remainder of the paper, e denotes the base of the natural logarithm.

The idea of the algorithm is simple. Early on, we should pick any item which arrives. As the knapsack fills, we should be more and more selective, that is, we pick items iff the value to weight ratio exceeds a certain function of the fraction of the knapsack filled. The precise function, in fact, can be obtained by solving a linear program, however, we only present the final algorithm for the sake of brevity.

Algorithm 1 ON-KP-Threshold

```
Let $\Psi(z) \equiv (Ue/L)^z(L/e)$.
When item $j$ arrives, let $z_j$ be the fraction of capacity filled, pick element $j$ iff $j$ doesn't overfill the knapsack and
    $\frac{v_j}{w_j} \geq \Psi(z_j)$.
```

The algorithm is presented in Algorithm 1 ON-KP-Threshold. Observe that for $z \in [0, c]$ where $c \equiv \frac{1}{1+\ln(U/L)}$, $\Psi(z) \leq L$, thus the algorithm will pick all items available until c fraction of the knapsack is filled. In fact, we will assume henceforth $\Psi(z) = L$ for $z \in [0, c]$. When $z = 1$, $\Psi(z) = U$, and since Ψ is strictly increasing, the algorithm will never over-fill the knapsack.

Theorem 2.1 ON-KP-Threshold *has a competitive ratio of* $\ln(U/L) + 1$.

Proof: Fix an input sequence σ. Let the algorithm terminate filling Z fraction of the knapsack and obtaining a value of $\mathcal{A}(\sigma)$. Let S and S^* respectively be the set of items picked by the Algorithm ON-KP-Threshold and the optimum. Denote the weight and the value of the common items by $W \equiv \sum_{j\in(S\cap S^*)} w_j$ and $P \equiv \sum_{j\in(S\cap S^*)} v_j$. For each item j not picked by the algorithm, its efficiency is $< \Psi(z_j) \leq \Psi(Z)$ since $\Psi(z)$ is a monotone increasing function of z. Thus,

$$\mathrm{OPT}(\sigma) \leq \mathrm{P} + \Psi(\mathrm{Z})(\mathrm{B} - \mathrm{W})$$

where $\mathrm{OPT}(\sigma)$ is the maximum value obtained by an offline algorithms given input σ. Let $v(S \setminus S^*) \equiv \sum_{j\in(S\setminus S^*)} v_j$, then $\mathcal{A}(\sigma) = P + v(S \setminus S^*)$. The above inequality implies that

$$\frac{\mathrm{OPT}(\sigma)}{\mathcal{A}(\sigma)} \leq \frac{P + \Psi(Z)(B - W)}{P + v(S \setminus S^*)}. \tag{1}$$

Since each item j picked in S must have efficiency at least $\Psi(z_j)$ where z_j is the fraction of the knapsack filled when the jth item arrives, we have

$$P \geq \sum_{j\in S\cap S^*} \Psi(z_j)w_j, \quad \text{call this } P_1 \tag{2}$$

$$v(S \setminus S^*) \geq \sum_{j\in S\setminus S^*} \Psi(z_j)w_j, \quad \text{call this } P_2. \tag{3}$$

Since $\mathrm{OPT}(\sigma) \geq \mathcal{A}(\sigma)$, Eq.(1) implies

$$\frac{\mathrm{OPT}(\sigma)}{\mathcal{A}(\sigma)} \leq \frac{P + \Psi(Z)(B - W)}{P + v(S \setminus S^*)} \leq \frac{P_1 + \Psi(Z)(B - W)}{P_1 + v(S \setminus S^*)} \leq \frac{P_1 + \Psi(Z)(B - W)}{P_1 + P_2} \tag{4}$$

By monotonicity of $\Psi()$, we get $P_1 \leq \Psi(Z)w(S \cap S^*) = \Psi(Z)W$. Note that $P_1 + P_2 = \sum_{j\in S} \Psi(z_j)w_j$. Plugging in the values of P_1 and P_2 we get

$$\frac{\mathrm{OPT}(\sigma)}{\mathcal{A}(\sigma)} \leq \frac{\Psi(Z)B}{\sum_{j\in S} \Psi(z_j)w_j} \leq \frac{\Psi(Z)}{\sum_{j\in S} \Psi(z_j)\Delta z_j} \tag{5}$$

where $\Delta z_j = z_{j+1} - z_j = w_j/B$ for all j.

Based on the assumption that the weights are much smaller than B, we can approximate the summation via an integration (refer to the remark following the proof). Thus,

$$\begin{aligned}
\sum_{j\in S}\Psi(z_j)\Delta z_j &\approx \int_0^Z \Psi(z)dz \\
&= \int_0^c L dz + \int_c^Z \Psi(z)dz \\
&= cL + \frac{L}{e}\frac{(Ue/L)^Z - (Ue/L)^c}{\ln(Ue/L)} \\
&= \frac{L}{e}\frac{(Ue/L)^Z}{\ln(Ue/L)} = \frac{\Psi(Z)}{\ln(U/L)+1}.
\end{aligned}$$

Along with Eq.(5), this completes the proof. □

Remark: For the above approximation equation, we rely on assumption 1 (the weights are much smaller than the capacity). To be precise, since $z_{j+1}-z_j \le \epsilon$, we get $\sum_{j\in S}\Psi(z_j)\Delta z_j \ge \int_0^{Z-\epsilon}\Psi(z)dz = \frac{\Psi(Z)}{\ln(U/L)+1}\cdot(Ue/L)^{-\epsilon}$. Thus, the competitive factor must be multiplied by $\texttt{error}(\epsilon) := (Ue/L)^{\epsilon}$ which tends to 1 as ϵ tends to 0. As mentioned in the introduction, we will assume $\epsilon = 0$ and avoid multiplying this factor everywhere to decrease notation.

2.1 A Matching Lower Bound

In this section we use Yao's minimax technique [13] to get a lower bound on the competitive ratio of the online knapsack problem, matching the upper bound given in Theorem 2.1.

Theorem 2.2 *The competitive ratio of any (possibly randomized) online algorithm for the online knapsack problem is at least* $(\ln(U/L)+1)$.

Proof: Yao's minimax principle says for any input distribution $\mathcal{D}$ and any γ-competitive randomized algorithm $\mathcal{A}$,

$$\frac{1}{\gamma} \le \min_{\sigma}\frac{\mathbf{E}[\mathcal{A}(\sigma)]}{\mathrm{OPT}(\sigma)} \le \max_{\text{deterministic } A} \mathbf{E}_{\sigma\leftarrow\mathcal{D}}\left[\frac{A(\sigma)}{\mathrm{OPT}(\sigma)}\right]$$

To prove the theorem we specify a distribution $\mathcal{D}$ such that

$$\max_{\text{deterministic } A} \mathbf{E}_{\sigma\leftarrow\mathcal{D}}\left[\frac{A(\sigma)}{\mathrm{OPT}(\sigma)}\right] \le \frac{1}{\ln(U/L)+1}. \tag{6}$$

Fix a parameter $\eta > 0$. Let k be the largest integer such that $(1+\eta)^k \le U/L$, i.e., $k = \lfloor\frac{\ln(U/L)}{\ln(1+\eta)}\rfloor$. The support of the input distribution $\mathcal{D}$ consists of the instances $I_0, I_1, \cdots, I_k$, where I_0 is a stream of B identical items each with weight 1 and value L. I_1 is I_0 followed by a stream of B identical items each with weight 1 and value $(1+\eta)L$, and in general I_{j+1} is I_j followed by B items with weight 1 and value $(1+\eta)^{j+1}L$. The distribution $\mathcal{D}$ is specified by giving probability p_j to instance I_j (we specify p_j's later).

Given knowledge of this distribution, any deterministic algorithm A can be fully specified by the vector $(f_0, f_1, \cdots, f_k)$, where f_i is the fraction of the knapsack it fills with items having efficiency ratio $(1+\eta)^i L$. Note that the f_j's could depend on the p_j's. Also note that $\sum_{j=0}^k f_j \le 1$ as the algorithm can not over-fill the knapsack. Thus we have

$$\mathbf{E}_{\sigma\leftarrow\mathcal{D}}\left[\frac{A(\sigma)}{\mathrm{OPT}(\sigma)}\right] = \sum_{i=0}^{k}\frac{p_i\sum_{j=0}^{i}(1+\eta)^j f_j}{(1+\eta)^i} = \sum_{j=0}^{k} f_j \sum_{i=j}^{k} p_i(1+\eta)^{j-i}$$

Now we specify the p_j's, $p_k \equiv \frac{1+\eta}{(k+1)\eta+1}$ and $p_0 = p_1 = \cdots = p_{k-1} \equiv \frac{\eta}{(k+1)\eta+1}$. Note that $\sum_j p_j = 1$. Let $X \equiv (k+1)\eta + 1$. For any j,

$$\begin{aligned}
\sum_{i=j}^{k} p_i(1+\eta)^{j-i} &= p_k(1+\eta)^{j-k} + \sum_{i=j}^{k-1} p_i(1+\eta)^{j-i} \\
&= \frac{(1+\eta)^{j-k+1}}{X} + \frac{\eta}{X}\sum_{i=j}^{k-1}(1+\eta)^{j-i} \\
&= \frac{(1+\eta)^{j-k+1}}{X} + \frac{\eta}{X}\left(\frac{(1+\eta)-(1+\eta)^{j-k+1}}{\eta}\right) \\
&= \frac{1+\eta}{X} = \frac{1+\eta}{(k+1)\eta+1}
\end{aligned}$$

Thus we get

$$\mathbf{E}_{\sigma\leftarrow\mathcal{D}}\left[\frac{\mathcal{A}(\sigma)}{\mathrm{OPT}(\sigma)}\right] = \frac{1+\eta}{(k+1)\eta+1}\sum_{j=0}^{k} f_j \leq \frac{1+\eta}{(k+1)\eta+1} \leq \frac{(1+\eta)}{\eta\frac{\ln(U/L)}{\ln(1+\eta)}+1}$$

where the inequality uses the fact that $\sum_{i=0}^{k} f_i \leq 1$ and $k+1 \geq \frac{\ln(U/L)}{\ln(1+\eta)}$. The proof completes by setting $\eta \to 0$ and noting that $\lim_{\eta\to 0}\frac{\eta}{\ln(1+\eta)} = 1$. $\square$

3 Extensions to Online Knapsack Variants

In this section we extend the algorithm for Online-KP to variants of online knapsack problems, particularly, the *multiple knapsack problem*, the *generalized assignment problem*, and the *multiple-choice knapsack problem.*

3.1 The Online Multiple Knapsack Problem

In the multiple knapsack problem, there are n knapsacks (or bins), with capacities $B_1, \cdots, B_n$. The values and weights of items are independent of the bins. For the online version (ON-MKP), one item arrives at one time and has to be either discarded or put into one of the bins.

The Algorithm ON-MKP-Threshold for ON-MKP works similarly to Algorithm 1. When an item j arrives, it is put into bin i if its efficiency satisfies the efficiency threshold of that bin at that instant (i.e., $v_j/w_j \geq \Psi(z_{ij})$), with ties broken arbitrarily. The proof of the following theorem is very similar to the proof of Theorem 2.1.

Theorem 3.1 ON-MKP-Threshold *is* $(\ln(U/L)+1)$*-competitive for* ON-MKP.

Proof: Let σ be a fixed input sequence and let the algorithm terminate filling $Z_1, Z_2, \cdots, Z_n$ fraction respectively of the n knapsacks obtaining a value of $\mathcal{A}(\sigma)$. Let S be the set of items picked by the algorithm and let $S_1, S_2, \cdots, S_n$ be the natural partition of S: S_i is set of items the algorithm picks in the ith knapsack.

Let $W_i \equiv \sum_{j\in(S_i\cap S^*)} w_j$ and $P_i \equiv \sum_{j\in(S_i\cap S^*)} v_j$. As before we have that for each item j not picked by the algorithm, its efficiency is $\leq \Psi(Z_i)$, for all i. Thus we get

$$\frac{\mathrm{OPT}(\sigma)}{\mathcal{A}(\sigma)} \leq \frac{\sum_{i=1}^{n}(\ P_i + \Psi(Z_i)(B_i - W_i)\ )}{\sum_{i=1}^{n}(\ P_i + v(S_i \setminus S^*)\ )}.$$

As in the proof of Theorem 2.1, we have for all i, $P_i \geq \sum_{j\in S_i\cap S^*}\Psi(z_{ij})w_j$ where z_{ij} is the fraction of the capacity of the ith knapsack filled when item j arrives. Thus,

$$\frac{\mathrm{OPT}(\sigma)}{\mathcal{A}(\sigma)} \leq \frac{\sum_{i=1}^{n}(\ \sum_{j\in S_i\cap S^*}\Psi(z_{ij})w_j + \Psi(Z_i)(B_i - W_i)\ )}{\sum_{i=1}^{n}(\ \sum_{j\in S_i\cap S^*}\Psi(z_{ij})w_j + v(S_i \setminus S^*)\ )}. \tag{7}$$

The numerator is less than $\sum_{i=1}^{n}(\Psi(Z_i)W_i + \Psi(Z_i)(B_i - W_i) = \sum_{i=1}^{n} \Psi(Z_i)B_i$.
The denominator is equal to $\sum_{i=1}^{n} \left(\sum_{j\in S_i} \Psi(z_{ij}) \cdot \Delta z_{ij}\right) B_i$, where $\Delta z_{ij} \equiv z_{i,j+1} - z_{ij} = w_j/B_i$.

As in the proof of Theorem 2.1, each of the ratios $(\Psi(Z_i)/\sum_{j\in S_i} \Psi(z_{ij})\Delta z_{ij})$ is less than $\frac{1}{(\ln(U/L)+1)}$ implying the RHS of Eq.(7) is less than $\frac{1}{(\ln(U/L)+1)}$. This completes the proof. □

Since KP is a special case of MKP, thus the lower bound of $\ln(U/L)+1$ in Theorem 2.2 also applies to ON-MKP. Therefore Algorithm ON-MKP-Threshold gives the optimal competitve ratio of $\ln(U/L)+1$ for ON-MKP.

3.2 The Online Generalized Assignment Problem

In the online generalized assignment problem (ON-GAP), there are n bins with capacities $B_1, \cdots, B_n$. At each time instant, an item j comes with value v_{ij} and weight w_{ij} for bin i. The goal is to allocate the item to a bin or discard it. The algorithm ON-GAP-Threshold works as follows: for each item, pick the bins which satisfy the threshold constraint of the single knapsack algorithm and assign it to the bin to which it gives the highest value.

Algorithm 2 ON-GAP-Threshold

Let $\Psi(z) \equiv (Ue/L)^z(L/e)$.
Whenever item j arrives, let $z_1, z_2, \cdots, z_n$ be the fraction of the bins filled already.

$$E_j \equiv \left\{ i \in [n] \mid \frac{v_{ij}}{w_{ij}} \geq \Psi(z_i) \right\},$$

allocate item j to the bin i in E_j with maximum v_{ij}, as long as it doesn't overshoot the capacity.

Theorem 3.2 *Algorithm* ON-GAP-Threshold *is* $(\ln(U/L)+2)$*-competitive for* ON-GAP.

Proof: Fix an input sequence σ of the item set Q. Suppose the algorithm ON-GAP-Threshold fills the bins to capacity $Z_1, Z_2, \cdots, Z_n$. Let $S_i, S_i^* \subset Q$ be the subset of items allocated by the algorithm and optimum respectively to bin i. Also let $S \equiv \cup_i S_i$ and $S^* \equiv \cup_i S_i^*$. Then $\mathrm{OPT}(\sigma) = \sum_\mathrm{i} \mathrm{v_i(S_i)}$ and $\mathcal{A}(\sigma) = \sum_i v_i(S_i^*)$.

We partition the set $S_i^* \setminus S$ into X_i^* and Y_i^*. X_i^* contains all items j which are not picked in S_i since its efficiency is less than $\Psi(z_{ij}) < \Psi(Z_i)$, z_{ij} be the fraction of the ith bin filled when the jth item arrived. Y_i^* are all those elements which satisfy the efficiency condition but is allocated to a different bin since it gives more value in that bin. Thus for all i, the items in Y_i^* are allocated by the algorithm but not in bin i.

Observe that the value obtained by the algorithm on allocating items of Y_i^* (call it $v_i(Y_i^*)$) is *more* than that obtained by the optimum algorithm. This is because the algorithm had an option of allocating these items to the bin to which the optimum algorithm allocated (that is i) but chose a more valuable bin instead. Thus,

$$\sum_{i=1}^{n} v_i(Y_i^*) \leq \sum_{i=1}^{n} v_i(S_i) = \mathcal{A}(\sigma). \tag{8}$$

As in the proof of Theorems 2.1 and 3.1, we have the following bound on the remaining items picked by the optimum algorithm,

$$\frac{\sum_{i=1}^{n}(v_i(S_i^*) - v_i(Y_i^*))}{\sum_{i=1}^{n} v_i(S_i)} \leq \frac{\sum_{i=1}^{n} \Psi(Z_i)B_i}{\sum_{i=1}^{n} \left(\sum_{j\in S_i} \Psi(z_{ij}) \cdot \Delta z_{ij}\right) B_i} \leq \ln(U/L) + 1.$$

Along with Eq.(8), we complete the proof. □

3.3 The Online Multiple-Choice Knapsack Problem

The online multiple choice knapsack problem (ON-MCKP) is a generalization of the online knapsack problem. At each time instant t, a set of items N_t arrives and the algorithm has to choose *at most* one item from the set. The goal again is to get as high a value without violating the capacity constraint of the knapsack. The algorithm for ON-MCKP is a simple modification of the algorithm ON-KP-Threshold: for every set N_t, find the elements which satisfy the efficiency threshold as per the online knapsack algorithm, and choose the one with maximum value.

Algorithm 3 ON-MCKP-Threshold

Let $\Psi(z) \equiv (Ue/L)^z(L/e)$.
At time t, let $z(t)$ denote the fraction of capacity filled,

$$E_t \equiv \left\{ s \in N_t \mid \frac{v_s}{w_s} \geq \Psi(z(t)) \right\},$$

pick element $s \in E_t$ with maximum v_s and add it to the knapsack as long as it doesn't overshoot the capacity.

The above algorithm has a competitive ratio of $\ln(U/L) + 2$, stated as the following theorem.

Theorem 3.3 *Algorithm* ON-MCKP-Threshold *has a competitive ratio of* $(\ln(U/L) + 2)$.

Proof: The proof is similar to that of Theorem 3.2. For any input sequence of sets σ, let $\mathcal{A}(\sigma)$ be the value obtained by the above algorithm and OPT(σ) be the maximum value obtainable. We claim that for any σ,

$$\text{OPT}(\sigma) - \mathcal{A}(\sigma) \leq (\ln(\text{U/L}) + 1)\mathcal{A}(\sigma).$$

Let S and S^* be the set of items picked by the algorithm and the optimum, respectively. Let $V = v(S \cap S^*)$ denote the value of the common items, $W = w(S \cap S^*)$ denote the weight. As before, we want to bound the value of the items picked by OPT but not by ALG. We partition the items picked by OPT and not by ALG into two: items which do not satisfy the efficiency condition, and the items which do. Thus the first kind of items have efficiency less than $\Psi(z(t))$, while for the second kind of items, the total value of these items is less than $\mathcal{A}(\sigma)$ since ALG picks the most valuable item from the same set which satisfy the efficiency condition. We can exclude the second type of items from further consideration since they in total result in at most a value of $\mathcal{A}(\sigma)$. Now we can assume that all items have efficiency $< \Psi(z(t))$ at time t, thus it returns to a similar situation as in the proof of Theorem 2.1. A similar proof shows that the above claim holds. □

4 Applications and Concluding Remarks

Since many real-world discrete decision problems can be modeled as variants of knapsack problems, we believe online algorithms we designed in this paper can be applied broadly. Particularly, we mention some applications to Internet advertising.

As an example, suppose we want to devise an online ad-serving algorithm for the auctioneer to maximize its revenue. There are a set of advertisers (bidders) while each of them has her own budget. Each time when a query arrives, advertisers will bid on different prices for this query and the auctioneer will select one ad to serve the user query. We can treat each bidder as a knapsack with its capacity as the bidder's budget. In such cases, the value (to the auctioneer, here) v_{ij} and the weight w_{ij} both equal to the bid of bidder i on query j. The problem can be modeled as an ON-GAP with $L = U = 1$. Algorithm ON-GAP-Threshold reduces to the following greedy algorithm: give the item to the highest bidder with sufficient budget remaining. The competitive ratio of this greedy algorithm is known to be 2 and this is implied by Theorem 3.2.

As another example, keyword auctions are used by Internet search engines like Google, Yahoo! and MSN to sell advertising positions in the search engine results page to advertisers. In fact, this work was motivated by this application and details can be found in [3]. For any given keyword, hundreds of bidders bid on it and are allowed to dynamically revise their bids. Usually there is a minimum bid b_{min} (specified by the system) for each keyword. At any moment of time a query is made for the keyword, the search engine allocates the highest S bidders (b_1 to b_S, say) to the S slots and displays their ads. The click-through-rate (CTR) of a slot s, denoted as $\alpha(s)$, is the probability for an ad on slot s to be clicked. If an advertisement on slot s is clicked, the advertiser is charged the bid of the bidder b_{s+1}.

Suppose we want to devise a bidding strategy for an advertiser in keyword auctions. The default advertiser usually associates a value-per-click V (eg. expected revenue) for the keyword. It also has a budget B over a time period T (24 hours if the budget is daily).

We discretize the time T into periods $\{1, 2, \cdots, T\}$ so that no bidder changes his bid in the time interval $[t, t+1)$. Let $X(t)$ denote the expected number of queries for the keyword in this interval. Thus, if at time t the advertiser bids minimum $p_s(t)$ to get slot s, thus she pays $p_s(t)X(t)\alpha(s)$ and his expected value is $(V - p_s(t))X(t)\alpha(s)$. Our problem can be stated thus: At each time t, bid appropriately to obtain position s, so as to maximize revenue while keeping total cost within budget. This can be modeled as an online multiple-choice knapsack problem (ON-MCKP) where at time t a set of items arrive:

$$N_t = \{s \in S \mid v_{ts} \equiv (V - p_s(t))X(t)\alpha(s), w_{ts} \equiv p_s(t)X(t)\alpha(s)\}.$$

If there are multiple keywords, we can model each keyword at each time interval as an itemset, while V_k denotes the expected value-per-click for keyword k. In the case where there is one ad slot in the search engine result page, it degenerates into the online knapsack problem (ON-KP). Bidding algorithms corresponding to online algorithms for ON-MCKP (or ON-KP) can be designed accordingly.

As a concluding remark, the upper and lower bounds of competitive ratios for both MCKP and GAP differ by 1, and it will be interesting to close the gap.

References

[1] N. Buchbinder and J. Naor. Online primal-dual algorithms for covering and packing problems. In *Proc. ESA*, pages 689–701, 2005.

[2] N. Buchbinder and J. Naor. Improved bounds for online routing and packing via a primal-dual approach. In *Proc. FOCS*, pages 293–304, 2006.

[3] D. Chakrabarty, Y. Zhou, and R. Lukose. Budget constrained bidding in keyword auctions and online knapsack problems. *Workshop on Sponsored Search Auctions*, 2007. Available at `http://www.hpl.hp.com/personal/Rajan_Lukose/papers/WWW2007-SSA.pdf`

[4] C. Chekuri and S. Khanna. A polynomial time approximation scheme for the multiple knapsack problem. *SIAM Journal on Computing*, 35(3):713–728, 2005.

[5] K. Iwama and S. Taketomi. Removable online knapsack problems. In *Proc. ICALP*, pages 293–305, 2002.

[6] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer, 2004.

[7] A. J. Kleywegt and J. D. Papastavrou. The dynamic and stochastic knapsack problem. *Operations Research*, 46(1):17–35, 1998.

[8] G. S. Lueker. Average-case analysis of off-line and on-line knapsack problems. *Journal of Algorithms*, 29(2):277–305, 1998.

[9] A. Marchetti-Spaccamela and C. Vercellis. Stochastic on-line knapsack problems. *Mathematical Programming*, 68:73–104, 1995.

[10] J. Noga and V. Sarbua. An online partially fractional knapsack problem. In *Proc. 8th Sym. Parallel Architectures, Algorithms and Networks*, pages 108–112, 2005.

[11] J. D. Papastavrou, S. Rajagopalan, and A. J. Kleywegt. The dynamic and stochastic knapsack problem with deadlines. *Management Science*, 42(12):1706–1718, 1996.

[12] R. van Slyke and Y. Young. Finite horizon stochastic knapsacks with applications to yield management. *Operations Research*, 48:155–172, 2000.

[13] A. C.-C. Yao. Probabilistic computations: towards a unified measure of complexity. In *Proc. FOCS*, pages 222–227, 1977.