On the Fully Polynomial Approximation Algorithm for the 0-1 Knapsack Problem

Yingjie Liu *

Abstract. A modified fast approximation algorithm for 0-1 knapsack problem with improved complexity is presented, based on the schemes of Ibarra, Kim and Babat. By using a new partition of items, the algorithm solves the n-item 0-1 knapsack problem to any relative error tolerance $\epsilon > 0$ in the scaled profit space $\frac{P^\star}{K} = O(\frac{1}{\epsilon^{1+\delta}})$ with time $O(n \log \frac{1}{\epsilon} + \frac{1}{\epsilon^{2+2\delta}})$ and space $O(n + \frac{1}{\epsilon^{2+\delta}})$, where $P^\star$ and b are the maximal profit and the weight capacity of the knapsack problem respectively, K is a problem-dependent scaling factor, $\delta = \frac{\alpha}{1+\alpha}$ and $\alpha = O(\log b)$. This algorithm reduces the exponent of the complexity bound in [5].

Key Words. Knapsack problem, subset-sum problem.

1 Introduction

The 0-1 knapsack problem (KP) is to solve for a given item set $N = \{1, 2, \cdots, n\}$,

$$\begin{aligned} P^\star &= \max \textstyle\sum_{j \in N} p_j x_j \\ \text{subject to} \quad & \textstyle\sum_{j \in N} w_j x_j \leq b, \end{aligned} \tag{1}$$

where $x_j \in \{0, 1\}$ and p_j, w_j are positive integers. KP is NP-hard, but it can be solved in pseudo-polynomial time $O(nb)$ through dynamic programming. Note that the space for storing input data b is $m = \log b$, therefore the complexity contains an exponential function of m. The first approximation scheme for KP was proposed by Sahni [9] and makes use of a greedy procedure.

*Department of Applied Mathematics and Statistics, SUNY at Stony Brook, NY 11789. email: yingjie@ams.sunysb.edu

The first fully polynomial approximation was found by Ibarra and Kim [3], and the same idea was also found by Babat [1] independently. The basic strategy in Ibarra-Kim algorithm is: (a)to separate items into a class of "large" items and a class of "small" items, according to the size of the profit p_j; (b)to solve the KP problem for the "large" items only, with profits scaled by a factor $K > 0$, through dynamic programming, and store the dynamic programming list in a table; (c)to fill the residual capacity for each entry in the table with the "small" items. The time and space complexity are $O(n \log n + \frac{1}{\epsilon^4} \log \frac{1}{\epsilon})$ and $O(n + \frac{1}{\epsilon^3})$ respectively. The Ibarra-Kim scheme has been modified by Lawler[5] to obtain improved time complexity $O(n \log \frac{1}{\epsilon} + \frac{1}{\epsilon^4})$ and space $O(n + \frac{1}{\epsilon^3})$, and by Magazine and Oguz [7] to obtain time complexity $O((n^2 \log n)/\epsilon)$ and space $O(\frac{n}{\epsilon})$, which uses smaller space when ϵ is set to be small enough, e.g., $O(1/n)$. In 1994, G. Gens and E. Levner [2] proposed an approximation algorithm for the subset-sum problem which has time complexity $O(\min(n/\epsilon, n + 1/\epsilon^3))$ and $O(\min(n/\epsilon, n + 1/\epsilon^2))$ space.

The main idea of this paper is to use a modified partitioning in the Ibarra-Kim scheme to reduce the size of the table described in (b), and then use the techniques in [5] to get scaled profit space $\frac{P^\star}{K} = O(\frac{1}{\epsilon^{1+\delta}})$, time complexity $O(n \log \frac{1}{\epsilon} + \frac{1}{\epsilon^{2+2\delta}})$ and space $O(n + \frac{1}{\epsilon^{2+\delta}})$ with $0 < \delta < 1$. In particular, $\delta = \frac{\alpha}{1+\alpha}$ with $\alpha = O(\log b)$. This is an improvement from [5] where the scaled profit space, time complexity and space are $\frac{P^\star}{K} = O(\frac{1}{\epsilon^2})$, $O(n \log \frac{1}{\epsilon} + \frac{1}{\epsilon^4})$ and $O(n + \frac{1}{\epsilon^3})$ respectively.

We first separate the items into "heavy", "large", and others, then solve the problem through dynamic programming for "heavy" and "Large" items only, with profits scaled by a factor K (differently for "heavy" and "Large" items). This yields a final list of (Q, A) pairs, where Q is the total scaled profit and A is the total weight satisfying $A \leq b$. The dominated pairs are discarded so that for any two pairs $(Q_1, A_1), (Q_2, A_2)$ in the list, either $Q_1 < Q_2, A_1 < A_2$ or $Q_1 > Q_2, A_1 > A_2$. For each pair (Q, A) in the list, fill the remaining capacity $b - A$ with other items using density greedy method (*i.e.*, ordering the items by non-increasing profit-to-weight ratios), which yields the total profit $\phi(b - A)$ for these items.

The following section provides two lemmas which are important for the proof of the main result in section 3.

2 Preparation

An item j is called "heavy" if $\frac{p_j}{\sqrt[\alpha]{w_j}} > D$; it is called "large" if $\sqrt[\alpha]{w_j} > W$ and $\frac{p_j}{\sqrt[\alpha]{w_j}} \leq D$. Where $D, W, \alpha > 0$ are to be determined later.

Lemma 1 *If the knapsack problem is calculated using arbitrary combination of "heavy" and "large" items only and with the remaining capacity filled with other items using density greedy method, then the optimal approximation profit P is bounded by $0 \leq P^\star - P \leq DW$.*

Proof: Suppose $P^\star$ is achieved by an item set $J \subseteq N$.
If J contains only "heavy" and "large" items, then $P = P^\star$; if not, suppose all "heavy" and "large" items in J consist a subset $I \subset J$. Note that $\sum_{i\in I} p_i + R \leq P$, where R is the total profit filling the capacity $b - \sum_{i\in I} w_i$ with other items through density greedy method. Also notice that $\sum_{j\in J-I} p_j - R \leq DW$ since the profit of every other item is no more than DW. Adding the above two inequalities we have $P^\star - P \leq DW$.

We scale the profit space with a factor $K > 0$, and the new profit q_j is defined as

$$q_j = \begin{cases} \lfloor \frac{p_j}{K} \rfloor & \text{if item } j \text{ is "heavy"}, \\ \lfloor \frac{p_j}{K} \rfloor + 1 & \text{if item } j \text{ is "large"}, \\ p_j & \text{otherwise.} \end{cases}$$

We have the following lemma.

Lemma 2 *If J is the set of "heavy" items and I is the set of "large" items, then*

$$-K|J| \leq K(\sum_{j\in I} q_j + \sum_{j\in J} q_j) - (\sum_{j\in I} p_j + \sum_{j\in J} p_j) \leq K|I|.$$

Proof:

$$\begin{aligned} &\tfrac{1}{K}\textstyle\sum_{j\in J} p_j - |J| \leq \sum_{j\in J} q_j \leq \tfrac{1}{K}\sum_{j\in J} p_j, \\ &\tfrac{1}{K}\textstyle\sum_{j\in I} p_j \leq \sum_{j\in I} q_j \leq \tfrac{1}{K}\sum_{j\in I} p_j + |I|, \end{aligned}$$

therefore,

$$(\sum_{j\in I} p_j + \sum_{j\in J} p_j) - K|J| \leq K(\sum_{j\in I} q_j + \sum_{j\in J} q_j) \leq (\sum_{j\in I} p_j + \sum_{j\in J} p_j) + K|I|.$$

3 Algorithm

Separate the items into three sets called "heavy", "large" and others, and solve the problem using "heavy" and "large" items only in the scaled profit space, then we get a final list of (Q, A) pairs. For each pair (Q, A) in the list, fill the remaining capacity $b - A$ with other items.

Let $P = \max_{(Q,A)} KQ + \phi(b - A)$. Suppose there exists an optimal solution (Q_o, A_o) in which the sum of the profits (unscaled) and the sum of the weights are P_H and A_H respectively for "heavy" items; P_L and A_L for "large" items. And P_s and A_s are the sum of the profits and the sum of the weights respectively for other items used to fill $b - A_o$. Let Q_H, Q_L be the sum of the scaled profits corresponding to P_H, P_L respectively, then there exists a pair (Q, A) in the final list s.t. $Q \geq Q_H + Q_L, A \leq A_H + A_L$. Therefore

$$P \geq KQ + \phi(b - A) \geq K(Q_H + Q_L) + \phi(b - A). \tag{2}$$

The number of "heavy" items contributing to the optimal cannot exceed $\frac{P^\star}{D\sqrt[\alpha]{w_0}} \leq \frac{P^\star}{D}$, where w_0 is the smallest weight which is no less than 1. Similarly, the number of "large" items cannot exceed $\frac{b}{W^\alpha}$. From Lemma 2 we have:

$$P^\star = P_H + P_L + P_s \leq K(Q_H + Q_L) + K\frac{P^\star}{D} + P_s. \tag{3}$$

Subtracting (2) from (3) gives:

$$P^\star - P \leq K\frac{P^\star}{D} + P_s - \phi(b - A). \tag{4}$$

From Lemma 1, it's easy to know that $P_s - \phi(b - A) \leq DW$, therefore

$$P^\star - P \leq K\frac{P^\star}{D} + DW. \tag{5}$$

We know P may be greater than $P^\star$ only because of the scaling of profit space, and we have an estimate of the number of "large" and "heavy" items for the pair (Q, A). In fact, from Lemma 2 and the computation of $\phi(b - A)$, we have

$$P - P^\star \leq K\frac{b}{W^\alpha}.$$

There exists a P_0 (it's determined by the corresponding linear programming problem in $O(n)$ time, for detail, see [5]), such that $P_0 \leq P^\star \leq 2P_0$. Therefore in order to have $|P - P^\star| \leq \epsilon P^\star$, we can set

$$\begin{cases} K\frac{2P_0}{D} + DW = \epsilon P_0, \\ K\frac{b}{W^\alpha} = \epsilon P_0. \end{cases} \tag{6}$$

Cancel out W from (6), we have

$$K\frac{2P_0}{D} + D(\frac{Kb}{\epsilon P_0})^{\frac{1}{\alpha}} = \epsilon P_0. \tag{7}$$

Assume

$$\frac{P_0}{K} \geq \frac{c}{\epsilon^{1+\delta}}, \tag{8}$$

where c is a constant, $c > 0$, $0 < \delta < 1$ are to be determined later. Since (8) implies $K \leq \frac{P_0\epsilon^{1+\delta}}{c}$, we get from (7) $K\frac{2P_0}{D} + D\epsilon^{\frac{\delta}{\alpha}} b^{\frac{1}{\alpha}} c^{-\frac{1}{\alpha}} \geq \epsilon P_0$. Solve for K we have

$$\frac{P_0}{K} \leq \frac{2}{\epsilon(\frac{D}{P_0}) - (\frac{D}{P_0})^2 \epsilon^{\frac{\delta}{\alpha}} b^{\frac{1}{\alpha}} c^{-\frac{1}{\alpha}}}. \tag{9}$$

Choose

$$D = \frac{\epsilon^{1-\frac{\delta}{\alpha}} c^{\frac{1}{\alpha}} P_0}{2b^{\frac{1}{\alpha}}}, \tag{10}$$

then the right hand side of (9) has a local minimum, we get

$$\frac{P_0}{K} \leq \frac{8b^{\frac{1}{\alpha}} c^{-\frac{1}{\alpha}}}{\epsilon^{2-\frac{\delta}{\alpha}}}. \tag{11}$$

Comparing (11) with (8) we can set $1 + \delta = 2 - \frac{\delta}{\alpha}$ and $c = 8b^{\frac{1}{\alpha}} c^{-\frac{1}{\alpha}}$, which implies

$$\alpha = \frac{\log b - \log c}{\log c - \log 8}, \ \delta = \frac{\alpha}{1+\alpha} \text{ and } \frac{P_0}{K} = \frac{c}{\epsilon^{1+\delta}}. \tag{12}$$

Finally, solve the second equation of (6), we get $W = (\frac{\epsilon^\delta b}{c})^{\frac{1}{\alpha}}$.

Theorem 1 *Let $c > 8$ be a constant. For any $\epsilon > 0$ and $b > c$, choose*

$$\alpha = \frac{\log b - \log c}{\log c - \log 8}, \ \delta = \frac{\alpha}{1+\alpha}, \ K = \frac{P_0 \epsilon^{1+\delta}}{c}, \ D = \frac{\epsilon^{1-\frac{\delta}{\alpha}} c^{\frac{1}{\alpha}} P_0}{2b^{\frac{1}{\alpha}}} \text{ and } W = (\frac{\epsilon^\delta b}{c})^{\frac{1}{\alpha}},$$

then $|P - P^\star| \leq \epsilon P^\star$, and $\frac{P^\star}{K} \leq \frac{2P_0}{K} = \frac{2c}{\epsilon^{1+\delta}}$.

Note that $\alpha = O(\log b)$, we can not obtain a problem-independent lower bound. However, it is interesting to see how large the profit space would be through some examples. In table 1, we calculate the size of δ with different values of c and b.

$b =$	10^2	10^3	10^4	10^5
$c = 16, \delta =$	0.726	0.856	0.903	0.927
$c = 32, \delta =$	0.451	0.713	0.806	0.853
$c = 64, \delta =$	0.177	0.569	0.708	0.780

Table 1: Comparative δ values for different b and c.

4 Remarks

Notice that $D/K = O(\frac{1}{\epsilon})$ according to theorem1, which is parallel to [5] with the partition parameter T replaced by D, therefore we can apply the techniques in [5] almost identically here, treating the "large" and "heavy" items indiscriminately. Therefore the time for computing the (Q, A) list of "large" and "heavy" items will be $O([\frac{P^\star}{K}] \cdot [\frac{P^\star}{K}]) = O(\frac{1}{\epsilon^{2+2\delta}})$. The time for computing $\phi(b - A)$ for all (Q, A) pairs in the final list will be $O(n \log(\frac{1}{\epsilon}) + [\frac{P^\star}{K}] \cdot \log(\frac{1}{\epsilon}))$. Therefore the total time will be $O(n \log \frac{1}{\epsilon} + \frac{1}{\epsilon^{2+2\delta}})$. Similarly the space will be $O(n + \frac{D}{K} \cdot [\frac{P^\star}{K}]) = O(n + \frac{1}{\epsilon^{2+\delta}})$.

Using the technique in [7] it is possible to trade more time for less space, we are not going to discuss the technical details here.

5 Acknowledgments

I would like to thank Dr. Ramesh Krishnamurti for his wonderful course in linear programming at Simon Fraser University, 1992. I became interested in the KP problem during the class and did a project on it, which is what this paper is based on. Also I would like to thank one referee for the valuable suggestions.

References

[1] L.G. Babat. *Linear functions on the N-dimensional unit cube.* Doklady Akademiia Nauk SSSR 222, 761-762, 1975.

[2] G. Gens and E. Levner. *A fast approximation algorithm for the subset-sum problem* INFOR, 32, pages 143-148, 1994.

[3] O.H. Ibarra , C.E. Kim. *Fast approximation algorithms for the Knapsack and sum of subset problems.* JACM V22, 4, pages 463-468, Oct.,1975.

[4] R. Kohli , R. Krishnamurti. *A total-value greedy heuristic for the integer knapsack problem.* Oper. Res. Lett. 12, pages 65-71, 1992.

[5] E.L. Lawler. *Fast approximation algorithm for knapsack Problem.* Proc. 18th Annual Symposium on Foundation of Computer Sci. IEEE Computer Society, Long Beach, pages 206-213, 1977.

[6] Y. Liu. *A fast approximation algorithm for the 0-1 Knapsack problem with adjustable complexity.* Graduate Course Project in Simon Fraser Univ., 1992.

[7] M.J. Magazine , O. Oguz. *A fully polynomial approximation algorithm for the 0-1 knapsack problem.* Europ. Journal of Oper. Res. 8, pages 270-273, 1981.

[8] S. Martello , P. Toth. *Algorithm for Knapsack Problems.* In Surveys in Combinatorial Optimization. North-Holland, pages 213-258, 1987.

[9] S. Sahni. *Approximate algorithms for the 0/1 Knapsack Problem.* JACM V22, 1, pages 115-124, Jan.,1975.