The Complexity of the 0/1 Multi-knapsack Problem

Zhang Li'ang (张立昂)
(Beijing Institute of Chemical Fibre Engineering)

Geng Suyun (耿素云)
(Beijing University)

Received March 1, 1983.

Abstract

In this paper complexity of the 0/1 multi-knapsack problem is discussed. First we prove that the corresponding decision problem is NP-complete in the strong sense. For any fixed number k of knapsacks, the problem is only NP-complete in the ordinary sense, but not NP-complete in the strong sense. Then, we prove that the 0/1 multi-knapsack optimization problem is NP-equivalent by using Turing reduction.

1. The Model

Suppose $A = \{a_i: 1 \leqslant i \leqslant n\}$, A is a multi-set if it is allowed that $a_i = a_j$ for $i \neq j$. The cardinal number of multi-set A is the number n of elements of A.

Consider the 0/1 multi-knapsack problem: Given multi-set $Q = \{(v_i, w_i): 1 \leqslant i \leqslant n\}$ and M_j, $1 \leqslant j \leqslant k$, where v_i, w_i and M_j are positive integers, find k disjoint subsets $J_j \subseteq \{1, 2, \cdots, n\}$, $1 \leqslant j \leqslant k$, so that

$$\sum_{i \in J_j} w_i \leqslant M_j, \quad 1 \leqslant j \leqslant k, \tag{1.1}$$

and

$$\sum_{j=1}^{k} \sum_{i \in J_j} v_i \text{ is maximum.} \tag{1.2}$$

The intuitive explanation of the 0/1 multi-knapsack problem is that we suppose n items and k knapsacks are given, where the weight of the ith item is w_i and its value is v_i, and total amount of weight of items allowed to be put into the jth knapsack is M_j, now ask which we should select among the n items and place them into the knapsacks so that the total value obtained reaches its maximum? J_j denotes the set of indexes of items placed into the jth knapsack. Obviously, we may generally assume that $w_i \leqslant \max\{M_j: 1 \leqslant j \leqslant k\}$, $1 \leqslant i \leqslant n$, and $M_j \leqslant \sum_{i=1}^{n} w_i$, $1 \leqslant j \leqslant k$. For $k = 1$, this is the ordinary 0/1 knapsack problem.

2. The Computing Complexity

We adopt the terms used in [1]. The 0/1 multi-knapsack problem is an optimization problem, and its corresponding decision problem can be described as follows:

2.1. 0/1 *multi-knapsack decision problem*

Instance: Multi-set $Q = \{(v_i, w_i): 1 \leqslant i \leqslant n\}$, $M_j \leqslant \sum_{i=1}^{n} w_i$, $1 \leqslant j \leqslant k$, and $V \leqslant \sum_{i=1}^{n} v_i$, where v_i, w_i, M_j, and V are positive integers.

Question: Are there k disjoint subsets $J_j \subseteq \{1, 2, \cdots, n\}$, $1 \leqslant j \leqslant k$, so that

$$\sum_{i \in J_j} w_i \leqslant M_j, \quad 1 \leqslant j \leqslant k, \tag{2.1}$$

and

$$\sum_{j=1}^{k} \sum_{i \in J_j} v_i \geqslant V. \tag{2.2}$$

This is a number problem, and its Length function and Max function are defined:

$$\text{Length}(I) = n + k + \max\{[\log_2 v_i], [\log_2 w_i] : 1 \leqslant i \leqslant n\},$$

and

$$\text{Max}(I) = \max\{v_i, w_i : 1 \leqslant i \leqslant n\},$$

where I is the instance of the problem.

Later we shall use another decision problem, which is NP-complete in the strong sense[1-3].

2.2. 3-*Partition*

Instance: A finite set $A = \{a_1, a_2, \cdots, a_{3m}\}$ of $3m$ elements, a positive integer bound B, and a positive integer "size" $s(a_i)$ so that for each $a_i \in A$, $s(a_i)$ satisfies $B/4 < s(a_i) < B/2$ and $\sum_{i=1}^{3m} s(a_i) = mB$.

Question: Can A be partitioned into m disjoint sets $A_1, A_2, \cdots, A_m$ so that for $1 \leqslant j \leqslant m$, $\sum_{a_i \in A_j} s(a_i) = B$? (Notice that the above constraints on the item sizes imply that every such A_j must contain exactly three elements.)

Its Length function and Max function might be defined as

$$\text{Length}'(I) = m + \max\{[\log_2 s(a_i)]: 1 \leqslant i \leqslant 3m\},$$

and

$$\text{Max}'(I) = \max\{s(a_i): 1 \leqslant i \leqslant 3m\}.$$

Theorem. *The* 0/1 *multi-knapsack decision problem is NP-complete in the strong sense, even if for* $1 \leqslant i \leqslant n$, $v_i = w_i$ *and for* $1 \leqslant i, j \leqslant k$, $M_j = M_i$.

Proof. It is easy to see that the problem is in NP, since for any given k disjoint subsets $J_j \subseteq \{1, 2, \cdots, n\}$, $1 \leqslant j \leqslant k$, we can check whether they satisfy condition (2.1) and (2.2) in polynomial time.

We will transform 3-Partition to the 0/1 multi-knapsack decision problem as follows. Let set $A = \{a_1, a_2, \cdots, a_{3m}\}$, positive integer "size" $s(a_i)$ for each $a_i \in A$, and a positive integer bound B constitute an arbitrary instance I of 3-Partition. The corresponding instance $f(I)$ of the 0/1 multi-knapsack decision problem is given by $Q = \{(s(a_i), s(a_i)): 1 \leqslant i \leqslant 3m\}$, $M_j = B$ for $1 \leqslant j \leqslant m$, $n = 3m$, $k = m$, and $V = mB$. We shall show that f is a pseudo-polynomial transformation.

First, if the answer for I is "Yes", that is, A can be partitioned into m disjoint subsets $A_1, A_2, \cdots, A_m$ so that $\sum_{a_i \in A_j} s(a_i) = B$ $(1 \leqslant j \leqslant m)$, then for $f(I)$, we take $J_j = \{i: a_j \in A_j\}$ $(1 \leqslant j \leqslant m)$. Thus, we have

(1) for $1 \leqslant j \leqslant k$, $\sum_{i \in J_j} w_i = \sum_{a_i \in A_j} s(a_i) = B \leqslant M_j$,

(2) $\sum_{j=1}^{k} \sum_{i \in J_j} v_i = \sum_{a_i \in A} s(a_i) = mB \geqslant V$.

This shows that the answer for $f(I)$ is also "Yes".

Conversely, suppose that the answer for $f(I)$ is "Yes", that is, there are k disjoint subsets $J_j \subseteq \{1, 2, \cdots, n\}$, $1 \leqslant j \leqslant k$, satisfying conditions (2.1) and (2.2). For I, we take $A_j = \{a_i : i \in J_j\}$ $(1 \leqslant j \leqslant m)$. From (2.2), we have $\sum_{j=1}^{m} \sum_{a_i \in A_j} s(a_i) = \sum_{j=1}^{k} \sum_{i \in J_j} v_i \geqslant V = mB$. Since $\sum_{i=1}^{3m} s(a_i) = mB$, A_1, A_2, $\cdots$, A_m must be a partition of A. Moreover, from (2.1), we have for $1 \leqslant j \leqslant m$, $\sum_{a_i \in A_j} s(a_i)$ $= \sum_{i \in J_j} w_i \leqslant M_j = B$, and since these m terms sum up to mB, so each $\sum_{a_i \in A_j} s(a_i)$ must be exactly B for $1 \leqslant j \leqslant m$. Hence the answer for I is also "Yes".

Thus we have shown that f satisfies condition (a) of the definition of pseudo-polynomial transformation, that is, for all I, the answer for I is "Yes" if and only if the answer for $f(I)$ is "Yes".

Secondly, since the construction of $f(I)$ is very simple, it is easy to see that f can be computed in time polynomial of Length$'(I)$, and so in time polynomial of the two variables Length$'(I)$ and Max$'(I)$ as well. Hence satisfies condition (b) of the difinition of pseudo-polynomial transformation.

Moreover, since for all I,

$$\text{Length}(f(I)) > \text{Length}'(I),$$

and

$$\text{Max}(f(I)) = \text{Max}'(I),$$

f must satisfy the other two conditions (c) and (d), that is, there exists a polynomial q_1 and a two-variable polynomial q_2 so that for all instance I of 3-Partition,

$$q_1(\text{Length}(f(I))) \geqslant \text{Length}'(I)$$

and

$$\text{Max}(f(I)) \leqslant q_2(\text{Max}'(I), \text{Length}'(I)).$$

Thus, we have shown that f is a pseudo-polynomial transformation from 3-Partition to the 0/1 multi-knapsack decision problem. Since 3-Partitions is NP-complete in the strong sense, the 0/1 multi-knapsack decision problem is also NP-complete in the strong sense. Notice that for all instance I of 3-Partition, $v_i = w_i$ $(1 \leqslant i \leqslant n)$ and $M_j = B$ $(1 \leqslant j \leqslant k)$ in our $f(I)$, hence the problem remains NP-complete in the strong sense even if for $1 \leqslant i \leqslant n$, $v_i = w_i$, and for $1 \leqslant i$, $j \leqslant k$, $M_j = M_i$.

When $k = 1$ and $v_i = w_i$ for $1 \leqslant i \leqslant n$, the 0/1 multi-knapsack problem becomes subset sum problem and remains NP-complete. It is easy to show that for any fixed positive integer k, the problem remains NP-complete even if we demand that $v_i = w_i$ for $1 \leqslant i \leqslant n$ and $M_j = M_i$ for $1 \leqslant i, j \leqslant k$. But for fixed k, one can easily construct a pseudo-polynomial time algorithm, which runs in time polynomial of two variables Max(I) and Length(I), hence the problem is not NP-complete in the strong sense for fixed k.

Now we are showing that the 0/1 multi-knapsack optimization problem is NP-equivalent. Suppose that there is a polynomial time algorithm for the 0/1 multi-knapsack problem. By using it we can construct a polynomial time algorithm for the corresponding decision problem as follows:

Find the optimal solution by using the algorithm for the optimization problem, calculate the value of $\sum_{j=1}^{k} \sum_{i \in J_j} v_i$, compare it with the given bound V, and then get the answer for the corresponding decision problem. Thus, the 0/1 multi-knapsack problem must be NP-hard. In

addition, it is not difficult to see that it is even NP-hard in the strong sence. Therefore, the problem cannot be solved in the polynomial time, nor even in the pseudo-polynomial time unless P = NP, so it is indeed harder than the ordinary 0/1 knapsack problem (assume that P ≠ NP).

In order to prove that the 0/1 multi-knapsack problem is also NP-easy, we introduce here an "intermediate problem", defined as follows:

3. 0/1 Multi-knapsack Replenishing Problem

Instance: A multi-set $Q = \{(v_i, w_i): 1 \leqslant i \leqslant n\}$, $M_j \leqslant \sum_{i=1}^{n} w_i$, $1 \leqslant j \leqslant k$, $V \leqslant \sum_{i=1}^{n} v_i$, and a "partial solution" $J_j \subseteq \{1, 2, \cdots, n\}$, $1 \leqslant j \leqslant k$, where v_i, w_i, M_j, and V are positive integers, and J_j, $1 \leqslant j \leqslant k$, are disjoint to each other.

Question: Can the partial solution be replenished to be a total solution, that is, do there exist k disjoint subsets $J_j' \subseteq \{1, 2, \cdots, n\}$, $1 \leqslant j \leqslant k$, so that

$$J_j \subseteq J_j', \quad 1 \leqslant j \leqslant k, \tag{3.1}$$

$$\sum_{i \in J_j'} w_i \leqslant M_j, \quad 1 \leqslant j \leqslant k, \tag{3.2}$$

$$\sum_{j=1}^{k} \sum_{i \in J_j'} v_i \geqslant V. \tag{3.3}$$

This problem is also NP-complete since the 0/1 multi-knapsack problem is its particular example in which $J_j = \phi$ for $1 \leqslant j \leqslant k$. We shall show that the 0/1 multi-knapsack optimization problem can be Turing-reduced to it as follows: Suppose $S[Q, M_1, \cdots, M_k, J_1, \cdots, J_k, V]$ is a subroutine for the 0/1 multi-knapsack replenishing problem. The parameters Q, M, M_i, J_j and V take the same meaning as in the instance. Let Q and $M_1, \cdots, M_k$ be an arbitrary instance of the 0/1 multi-knapsack problem. First we compute the optimal value V^* by calling $S[Q, M_1, \cdots, M_k, \phi, \cdots, \phi, V]$ with different values of V. Notice that $V^* \leqslant \sum_{i=1}^{n} v_i$, we need at most $[\log_2 \sum_{i=1}^{n} v_i]$ calls of the subroutine with the binary search technique. Then we execute

Step 1. Set $J_j \leftarrow \phi, 1 \leqslant j \leqslant k$, and $i \leftarrow 1$.

Step 2. Set $j \leftarrow 1$.

Step 3. Call $S[Q, M_1, \cdots, M_k, J_1, \cdots, J_{j-1}, J_j \cup \{i\}, J_{j+1}, \cdots, J_k, V^*]$. If the answer is "Yes", then set $J_j \leftarrow J_j \cup \{i\}$ and go to Step 5.

Step 4. If $j < k$, then set $j \leftarrow j + 1$ and go to Step 3.

Step 5. If $i < n$, then set $i \leftarrow i + 1$ and go to Step 2.

It is easy to see that this procedure is indeed an algorithm for the 0/1 multi-knapsack problem. In Step 3. at most kn calls of the subroutine should be made and the algorithm at most totally requires $kn + [\log_2 \sum_{i=1}^{n} v_i]$ calls of the subroutine. Notice that the size of instance of the replenishing problem changes not greatly in each call and except the calls of the subroutine the amount of computation is quite small. Hence, if the subroutine S is a polynomial time algorithm, then this algorithm would be a polynomial time one. Therefore, the algorithm is a Turing reduction from the 0/1 multi-knapsack problem to the 0/1 multi-knapsack replenishing problem.

Thus we conclude that the 0/1 multi-knapsack problem is NP-equivalent, and that it is solvable in polynomial time if and only if P = NP.

References

[1] M.R.Garey and D.S.Johnson, Â Guide to the Theory of NP-Completeness, W.H.Freeman and Company, San Francisco, 1979 .

[2] M.R.Garey and D.S.Johnson, Complexity Results for Multiprocessor Scheduling under Resource Constraints, *SIAM J. Computing*, **4**(1975), 397—411.

[3] M.R.Garey and D.S.Johnson, Strong NP-Completeness Results: Motivation, Examples, and Implications, *J. Assoc.Comput. Mach.*, **25**(1978), 499—508.