Journal of the Operations Research
Society of Japan
Vol.21, No.1, March, 1978

THE MULTIPLE-CHOICE KNAPSACK PROBLEM

Toshihide Ibaraki and Toshiharu Hasegawa

Department of Applied Mathematics and Physics, Faculty of Engineering

Kyoto University, Kyoto, Japan

Katsumi Teranaka

Yokosuka Electrical Communication Laboratory, N.T.T., Yokosuka, Japan

and

Jiro Iwase

IBM Japan, Ltd., Osaka, Japan

(Received April 12, 1977; and Revised August 2, 1977)

ABSTRACT

This paper treats the multiple-choice (continuous) knapsack problem P:
maximize $\sum_{i=1}^{n} \sum_{j=1}^{m_i} c_{ij} x_{ij}$ subject to (1) $\sum_{i=1}^{n} \sum_{j=1}^{m_i} a_{ij} x_{ij} \leqslant b$, (2) $0 \leqslant x_{ij} \leqslant 1$, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m_i$ and (3) at most one of $x_{i1}, x_{i2}, \ldots, x_{im_i}$ is positive for $i = 1, 2, \ldots, n$, where n, m_i are positive integers and a_{ij}, c_{ij}, b are nonnegative real numbers. Two approximate algorithms and an exact branch-and-bound algorithm are proposed, by making use of the property that the LP relaxation of P provides considerably accurate upper and lower bounds of the optimal value of P. Although the multiple-choice knapsack problem is known to be NP-complete, computation results are quite encouraging. For example, approximate solutions withing 0.001% of the optimal values are obtained in less than one second (on FACOM 230/60) for problems with $n = 1000$ and $m_i = 2$, which are randomly generated from the uniform distribution. Exact optimal solutions of these problems with $n = 500$ and $m_i = 2$ are also obtained in less than 0.2 seconds (on FACOM M190).

1. Introduction

A variant of the well known knapsack problem is discussed in this paper.

$$P: \text{maximize } z=\sum_{i=1}^{n}\sum_{j=1}^{m_i} c_{ij}x_{ij} \tag{1.1}$$

$$\text{subject to } \sum_{i=1}^{n}\sum_{j=1}^{m_i} a_{ij}x_{ij}\leq b \tag{1.2}$$

$$0\leq x_{ij}\leq 1,\ i=1,2,\ldots,n,\ j=1,2,\ldots,m_i \tag{1.3}$$

$$\text{At most one of } x_{i1}, x_{i2},\ldots, x_{im_i} \text{ is positive, for } i=1,2,\ldots,n, \tag{1.4}$$

where n, m_i are positive integers, a_{ij}, b are nonnegative real numbers, and c_{ij} are real numbers. This problem P is called the *multiple-choice (continuous) knapsack problem* since it requires to select at most one item from each of n groups (group i has m_i different items). The amount x_{ij} of each selected item is measured by a real number between 0 and 1.

For example, suppose that there are n different types of space foods to be loaded on a satellite. There are m_i different brands in each type, and at most one of them is selected, where each brand has its own weight and value (measured by a real number). Our problem is to decide types, brands and their amount to be loaded on a satellite so that the total value is maximized under the total weight constraint.

Although this problem has apparently not been treated in the literature, a discrete version with the additional constraint $x_{ij}=0$ or 1 was investigated by Chandra, Hirschberg and Wong [2], and Nauss [16]. As we shall see, some of their results are extensible to our problem. In case of $m_i=2$ for every i, the problem P becomes a special case of the complementary programming problem [5]. Our first motivation was to use P as a relaxation problem to obtain upper bounds in a branch-and-bound algorithm for the general complementary programming problem.

It is emphasized here that the multiple-choice constraint (1.4) (with $m_i>1$) and the upper bound on x_{ij} in (1.3)† are crucial from the view point of computational complexity, since P can be easily solved if $m_i=1$ for $i=1,2,\ldots,n$ (then P is the ordinary *continuous* knapsack problem) or if x_{ij} is only bounded below (then an optimal solution is given by $x_{\bar{i}\bar{j}}=b/a_{\bar{i}\bar{j}}$ where $x_{\bar{i}\bar{j}}$ has

† The particular value 1 of the upper bound does not lose generality since $0\leq x_{ij}\leq d_{ij}$ can be transformed to $0\leq X_{ij}\leq 1$ by letting $X_{ij}=x_{ij}/d_{ij}$.

the maximum c_{ij}/a_{ij}). In the above general setting (even if $m_i=2$), the multiple-choice knapsack problem is known to be NP-complete [9]. The NP-completeness strongly suggests that there exists no algorithm which always runs in computation time bounded by a polynomial of the length of input data, i.e., n, m and $\log b$. (For further implication of the NP-completeness, see [1] [12] for example.)

To avoid this computational difficulty, we propose two approximate algorithms in Sections 5 and 6, after discussing some properties of the LP (Linear Programming) relaxation $\bar{P}$ (i.e., P with constraints (1.3) and (1.4) replaced by $0 \leq \sum_{j=1}^{m_i} x_{ij} \leq 1$, $i=1,2,\ldots, n$) and its dual $\bar{D}$ in Sections 3-4. These approximate algorithms run in polynomial time, and their performance seems to be extremely good. As reported in Section 7, for example, approximate solutions within 0.001% of the optimal values are obtained in less than one second on FACOM 230/60 for problems with $n=1000$ and $m_i=2$ for every i which are randomly generated from the uniform distribution. This should be sufficient for practical purposes.

In Section 8, we then construct a branch-and-bound algorithm for obtaining exact optimal solutions, by making use of the LP relaxation $\bar{P}$ and the above approximate solutions. Its computational results are also good as reported in Section 9. For example, problems with $n=500$ and $m_i=2$ for every i, which are randomly generated from the uniform distribution, are solved in less than 0.2 seconds on FACOM M190. The computation time seems to be $O(n \log n)$, as opposed to the NP-completeness result. Therefore, we also test highly structured difficult problems. The computation time for these problems seems to grow exponentially with n.

In conclusion, we may say that the multiple-choice knapsack problem is rather easy in the sense of the average computation time (not the worst case time), among a variety of NP-complete problems.

2. Some Simplification of P

In the definition of P in Section 1, it was assumed that $a_{ij} \geq 0$. This loses some generality of the problem, but it is satisfied in most cases practically encountered. We now give some further assumptions which do not lose generality but simplify the subsequent discussion.

First it can be assumed without loss of generality that

$$c_{ij} > 0, \quad i=1,2,\ldots, n, \quad j=1,2,\ldots, m_i, \tag{2.1}$$

since $c_{ij} \leq 0$ implies that $x_{ij}=0$ can be assumed in optimal solutions (thus x_{ij} can be deleted from the formulation of P). Also

$$(2.2) \qquad \begin{aligned} &a_{i1} \geq a_{i2} \geq \ldots \geq a_{im_i} \\ &c_{i1} \geq c_{i2} \geq \ldots \geq c_{im_i}, \quad i=1,2,\ldots, n \end{aligned}$$

does not lose generality. To prove this, first rearrange the second subscripts (if necessary) so that $a_{i1} \geq a_{i2} \geq \ldots \geq a_{im_i}$ holds. Assume $a_{ik} \geq a_{ik+1}$, $c_{ik} < c_{ik+1}$, $x'_{ik}=\alpha$ (>0) and $x'_{ik+1}=0$ hold in an optimal solution x'. Then the new solution $x''_{ik}=0$, $x''_{ik+1}=\alpha$ (all other variables do not alter their values) also satisfies constraints (1.2)-(1.4), but its objective value of x_{ik} and x_{ik+1} is

$$\begin{aligned} c_{ik}x''_{ik}+c_{ik+1}x''_{ik+1} &= c_{ik+1}\alpha \\ &> c_{ik}\alpha = c_{ik}x'_{ik}+c_{ik+1}x'_{ik+1}. \end{aligned}$$

This contradicts the optimality of x'. Thus $x_{ik}=0$ can be assumed and x_{ik} can be deleted from P.

The next assumption is

$$(2.3) \qquad \frac{c_{im_i}}{a_{im_i}} > \frac{c_{im_i-1}}{a_{im_i-1}} > \ldots > \frac{c_{i1}}{a_{i1}}, \quad i=1,2,\ldots, n.$$

(Here we use the conventions that $\frac{c}{0} > \frac{c'}{a}$ for any c, c', $a>0$, and $\frac{c}{0} \geq \frac{c'}{0}$ if and only if $c \geq c'$.)

This also does not lose generality as proved below. Let $a_{ik} \geq a_{ik+1}$ and $c_{ik} \geq c_{ik+1}$, and assume that

$$\frac{c_{ik}}{a_{ik}} \geq \frac{c_{ik+1}}{a_{ik+1}} .$$

If an optimal solution x' satisfies $x'_{ik}=0$ and $x'_{ik+1}=\alpha$ (>0) consider the new solution x'' with $x''_{ik}=(a_{ik+1}/a_{ik})\alpha$ (=α if $a_{ik+1}=a_{ik}=0$), $x''_{ik+1}=0$ (all other variables do not alter their values). It is easy to show that x'' is also feasible in P and has a greater or equal objective value. Thus $x_{ik+1}=0$ can be assumed without losing all optimal solutions, and hence x_{ik+1} can be deleted from P.

As a result of (2.2), we can assume without loss of generality that

$$(2.4) \qquad \sum_{i=1}^{n} a_{i1} > b ,$$

Since otherwise the solution $x_{11}=x_{21}=\ldots=x_{n1}=1$ and $x_{ij}=0$ for $j>1$ is trivially

optimal.

Now we give a simple property of optimal solutions of P.

Theorem 2.1. P (defined by (1.1)-(1.4)) has an optimal solution such that all variables x_{ij} except at most one satisfy $x_{ij}=0$ or 1.

Proof: Assume that an optimal solution x' satisfies $x'_{ij}=\alpha$, $x'_{i'j'}=\beta$, where $0<\alpha, \beta<1$. Assume $(c_{ij}/a_{ij})\geq(c_{i'j'}/a_{i'j'})$ without loss of generality. Consider the new solution x'':

$$x''_{ij}=1,\ x''_{i'j'}=\beta-(a_{ij}/a_{i'j'})(1-\alpha) \text{ if } \beta-(a_{ij}/a_{i'j'})(1-\alpha)\geq 0,$$

$$x''_{ij}=\alpha+(a_{i'j'}/a_{ij})\beta,\ x''_{i'j'}=0 \text{ otherwise}$$

(all other variables do not alter their values). x'' is feasible in P and has a greater objective value, as easily proved, a contradiction. □

This suggests that our problem P differs only slightly from its discrete version P' discussed in [2]. Sinse P seems computationally easier than P' as will be shown in the subsequent discussion, it may be possible to use optimal solutions of P to construct approximate solutions of P' or to obtain upper bounds in branch-and-bound algorithms for solving P'.

3. LP Relaxation of P and Its Dual

The LP (Linear Programming) relaxation $\bar{P}$ of P is introduced in this section in preparation for the subsequent discussion.

$$\bar{P}: \text{maximize } z=\sum_{i=1}^{n}\sum_{j=1}^{m_i} c_{ij}x_{ij} \tag{3.1}$$

$$\text{subject to } \sum_{i=1}^{n}\sum_{j=1}^{m_i} a_{ij}x_{ij}\leq b \tag{3.2}$$

$$\sum_{j=1}^{m_i} x_{ij}\leq 1, \quad i=1,2,\ldots,n \tag{3.3}$$

$$x_{ij}\geq 0, \quad i=1,2,\ldots,n, \quad j=1,2,\ldots,m_i. \tag{3.4}$$

This is an LP problem with a feasible region greater than P. As will be shown in Section 4, an optimal solution of $\bar{P}$ can be easily obtained (without using the simplex method) by considering its LP dual $\bar{D}$.

$$\bar{D}: \text{minimize } v=\sum_{i=1}^{n}\mu_i+b\lambda \tag{3.5}$$

(3.6) subject to $\mu_i \geq c_{ij} - a_{ij}\lambda, \quad i=1,2,\ldots,n, \quad j=1,2,\ldots,m_i$

(3.7) $\mu_i \geq 0, \quad i=1,2,\ldots,n$

(3.8) $\lambda \geq 0.$

In the rest of this section, we give an algorithm to obtain an optimal solution of $\bar{D}$. Let λ be fixed to a certain nonnegative value, and denote the resulting problem by $\bar{D}(\lambda)$. Since each μ_i can be independently determined, the optimal value $\bar{v}(\lambda)$ of $\bar{D}(\lambda)$ is given by

(3.9) $$\bar{v}(\lambda) = \sum_{i=1}^{n} \bar{\mu}_i(\lambda) + b\lambda$$

(3.10) $$\bar{\mu}_i(\lambda) = \max[0, \max\{c_{ij} - a_{ij}\lambda \mid j=1,2,\ldots,m_i\}], \quad i=1,2,\ldots,n.$$

Thus $\bar{D}$ is solved by finding a λ that minimizes $\bar{v}(\lambda)$.

Now note that two lines (in λ-ν plane)

(3.11) $$\begin{aligned} \nu &= c_{ij} - a_{ij}\lambda \\ \nu &= c_{ij-1} - a_{ij-1}\lambda \end{aligned}$$

cross at

(3.12) $$\lambda = \frac{c_{ij-1} - c_{ij}}{a_{ij-1} - a_{ij}}.$$

By direct manipulation, it follows from (2.2) and (2.3) that coefficients (3.12) satisfy

(3.13) $$\frac{c_{ij}}{a_{ij}} > \frac{c_{ij-1}}{a_{ij-1}} > \frac{c_{ij-1} - c_{ij}}{a_{ij-1} - a_{ij}}.$$

It can be further assumed that

(3.14) $$\frac{c_{i1} - c_{i2}}{a_{i1} - a_{i2}} < \frac{c_{i2} - c_{i3}}{a_{i2} - a_{i3}} < \ldots < \frac{c_{im_i-1} - c_{im_i}}{a_{im_i-1} - a_{im_i}}, \quad i=1,2,\ldots,n$$

(the same conventions as stated after (2.3) are assumed) as far as optimal solutions of $\bar{D}$ (and hence $\bar{P}$) are concerned. To show this, let

(3.15) $$\frac{c_{ij-1} - c_{ij}}{a_{ij-1} - a_{ij}} \leq \frac{c_{ij-2} - c_{ij-1}}{a_{ij-2} - a_{ij-1}}$$

hold for some i and j. Then by (2.2) and (2.3)

$$\max\{c_{ij}-a_{ij}\lambda,\ c_{ij-2}-a_{ij-2}\lambda\} \geq c_{ij-1}-a_{ij-1}\lambda$$

always holds as illustrated in Fig. 1. This means that the constraint $\mu_i \geq c_{ij-1}-a_{ij-1}\lambda$ (i.e., coefficients c_{ij-1}, a_{ij-1}) can be deleted from $\bar{D}$ since $\bar{\mu}_i(\lambda) < c_{ij-1}-a_{ij-1}\lambda$ never holds for feasible μ_i and x. Deleting these dummy constraints and adjusting subscripts, we obtain $\bar{D}$ satisfying (3.14).

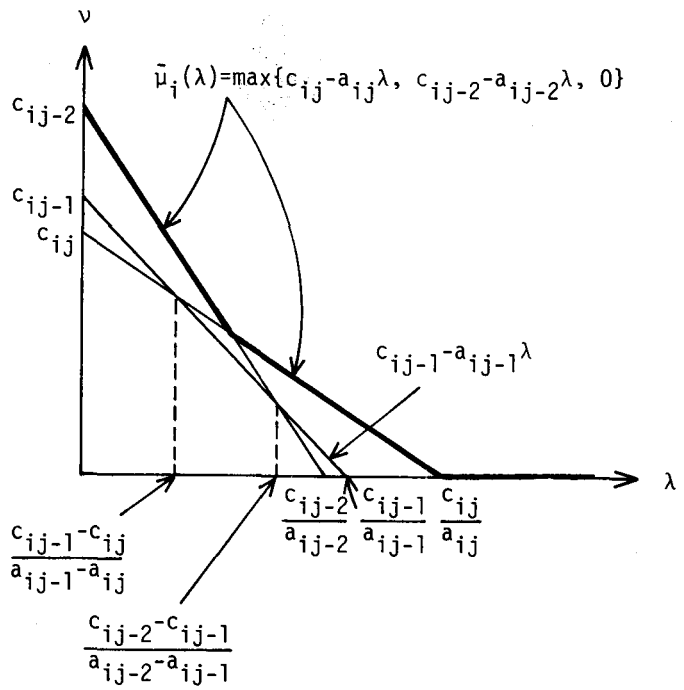

Fig. 1. Illustration of lines $\nu=c-a\lambda$ when $(c_{ij-1}-c_{ij})/(a_{ij-1}-a_{ij}) < (c_{ij-2}-c_{ij-1})/(a_{ij-2}-a_{ij-1})$ holds.

The property (3.14) was also proved in [2]. However, their proof is more complicated since [2] does not use the concept of $\bar{D}$. It should also be noted that (3.14) cannot be generally assumed in P. This sometimes introduces some complication in algorithms for solving P (not $\bar{P}$).

Now let $\bar{D}$ satisfy (2.1)-(2.4) and (3.14). Define $j_i(\lambda)$ for $\lambda \geq 0$ as follows (see Fig. 2).

$$j_i(\lambda)=1 \quad \text{if } 0 \leq \lambda < (c_{i1}-c_{i2})/(a_{i1}-a_{i2})$$

(3.16)
$$=j-1 \text{ or } j \text{ if } \lambda=(c_{ij-1}-c_{ij})/(a_{ij-1}-a_{ij})$$

$$=j-1 \quad \text{if } j\geq 3 \text{ and } (c_{ij-2}-c_{ij-1})/(a_{ij-2}-a_{ij-1})<$$

$$\lambda<(c_{ij-1}-c_{ij})/(a_{ij-1}-a_{ij}) \quad (\text{or } c_{im_i}/a_{im_i} \text{ if } j-1=m_i)$$

(3.16)
$$=m_i \text{ or } \infty \text{ if } \lambda=c_{im_i}/a_{im_i}$$

$$=\infty \quad \text{if } c_{im_i}/a_{im_i}<\lambda.$$

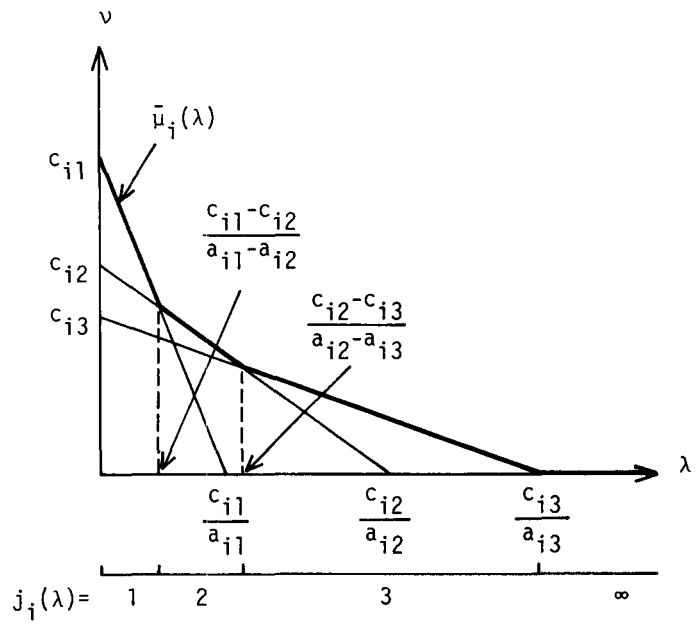

Fig. 2. Illustration of $j_i(\lambda)$ and $\bar{\mu}_i(\lambda)$ under assumptions (2.2), (2.3), (2.4) and (2.14).

Then obviously

(3.17) $$\bar{\mu}_i(\lambda)=c_{ij_i(\lambda)}-a_{ij_i(\lambda)}\lambda, \quad i=1,2,\ldots, n$$

holds, where

(3.18) $$c_{i\infty}=a_{i\infty}=0$$

is assumed for convenience. This gives

$$\bar{v}(\lambda) = \sum_{i=1}^{n} (c_{ij_i(\lambda)} - a_{ij_i(\lambda)}\lambda) + b\lambda \tag{3.19}$$

and hence

$$\frac{d\bar{v}}{d\lambda} = b - \sum_{i=1}^{n} a_{ij_i(\lambda)} \tag{3.20}$$

holds except for the points $\lambda = c_{im_i}/a_{im_i}$ and $\lambda = (c_{ij-1} - c_{ij})/(a_{ij-1} - a_{ij})$ at which $\bar{v}(\lambda)$ is not differentiable.

First let $\lambda = \infty$ (i.e., sufficiently large) and $j_i(\lambda) = \infty$ for all i, then

$$\left.\frac{d\bar{v}}{d\lambda}\right|_{\lambda=\infty} = b \ (\geq 0).$$

Then decrease λ continuously to $\lambda = 0$. From (3.20) and the definition of $j_i(\lambda)$, we see that $d\bar{v}/d\lambda$ decreases by the amount

a_{im_i} when λ crosses c_{im_i}/a_{im_i}, and

$a_{ij-1} - a_{ij}$ (≥ 0) when λ crosses $(c_{ij-1} - c_{ij})/(a_{ij-1} - a_{ij})$.

When $\lambda = 0$ is reached,

$$\left.\frac{d\bar{v}}{d\lambda}\right|_{\lambda=0} = b - \sum_{i=1}^{n} a_{i1} < 0 \tag{3.21}$$

holds by (2.4). Thus $\bar{v}(\lambda)$ changes as illustrated in Fig. 3. We want to find λ which minimizes $\bar{v}(\lambda)$, i.e., $\lambda = \bar{\lambda}$ such that

$$\left.\frac{d\bar{v}}{d\lambda}\right|_{\lambda=\bar{\lambda}+0} \geq 0 \text{ and } \left.\frac{d\bar{v}}{d\lambda}\right|_{\lambda=\bar{\lambda}-0} \leq 0. \tag{3.22}$$

To find this $\bar{\lambda}$, first compute

$$\frac{c_{im_i}}{a_{im_i}}, \frac{c_{im_i-1} - c_{im_i}}{a_{im_i-1} - a_{im_i}}, \ldots, \frac{c_{i2} - c_{i3}}{a_{i2} - a_{i3}}, \frac{c_{i1} - c_{i2}}{a_{i1} - a_{i2}}, \quad i = 1, 2, \ldots, n,$$

and arrange these in nondecreasing order, i.e.,

$$\beta_1 \leq \beta_2 \leq \ldots \leq \beta_N, \tag{3.23}$$

where

$$N = \sum_{i=1}^{n} m_i. \tag{3.24}$$

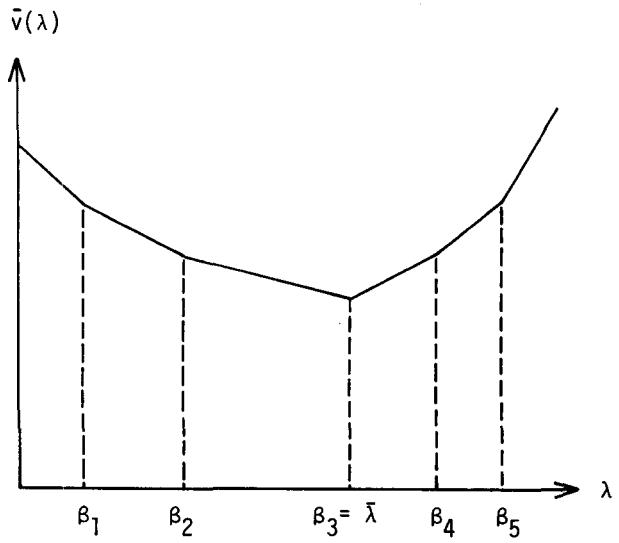

Fig. 3. Illustration of $\bar{v}(\lambda)$ and $\bar{\lambda}$.

Note that β_k is of the form either $\beta_k=c_{im_i}/a_{im_i}$ or $\beta_k=(c_{ij-1}-c_{ij})/(a_{ij-1}-a_{ij})$. This sorting requires $O(N \log N)$ computation time (e.g., [1] [13]). The ordered list containing β_k's is denoted by L.

Now the following algorithm obtains an optimal solution of $\bar{D}$. v' is used to denote $d\bar{v}/d\lambda$.

Algorithm DUAL(P)

D1 : Obtain list $L=\{\beta_1, \beta_2, \ldots, \beta_N\}$, where β_k's are sorted as in (3.23).

D2 : $v' \leftarrow b$, $j_i \leftarrow \infty$ for $i=1,2,\ldots, n$, $k \leftarrow N$.

D3 :

$$v' \leftarrow \begin{cases} v'-a_{im_i} & \text{if } \beta_k=c_{im_i}/a_{im_i} \\ v'-(a_{ij-1}-a_{ij}) & \text{if } \beta_k=(c_{ij-1}-c_{ij})/(a_{ij-1}-a_{ij}) \end{cases}$$

$$j_i \leftarrow \begin{cases} m_i & \text{if } \beta_k=c_{im_i}/a_{im_i} \\ j-1 & \text{if } \beta_k=(c_{ij-1}-c_{ij})/(a_{ij-1}-a_{ij}) \end{cases}$$

(Other j_i's do not change)

D4 : If $v' \leq 0$, go to D5; else return to D3 after letting $k \leftarrow k-1$.

D5 : $\bar{k}\leftarrow k$, $\bar{\lambda}\leftarrow\beta_{\bar{k}}$

$$\bar{e}\leftarrow\begin{cases} v'+a_{\bar{i}m_{\bar{i}}} & \text{if } \beta_{\bar{k}}=c_{\bar{i}m_{\bar{i}}}/a_{\bar{i}m_{\bar{i}}} \\ v'+a_{\bar{i}\bar{j}-1} & \text{if } \beta_{\bar{k}}=(c_{\bar{i}\bar{j}-1}-c_{\bar{i}\bar{j}})/(a_{\bar{i}\bar{j}-1}-a_{\bar{i}\bar{j}}), \end{cases}$$

and halt. □

Denote j_i's computed by DUAL(P) by $j_i(\bar{k})$. Then $j_i(\bar{k})$ and $\bar{e}$ satisfy

$$\begin{aligned} j_i(\bar{k})&=1 \quad \text{if } \bar{k}\le p \text{ where } \beta_p=(c_{i1}-c_{i2})/(a_{i1}-a_{i2}) \\ &=j-1 \quad \text{if } p<\bar{k}\le q \text{ where } \beta_p=(c_{ij-2}-c_{ij-1})/(a_{ij-2}-a_{ij-1}) \\ &\qquad \text{and } \beta_q=(c_{ij-1}-c_{ij})/(a_{ij-1}-a_{ij}) \\ &=m_i \quad \text{if } p<\bar{k}\le q \text{ where } \beta_p=(c_{im_i-1}-c_{im_i})/(a_{im_i-1}-a_{im_i}) \text{ and} \\ &\qquad \beta_q=c_{im_i}/a_{im_i} \\ &=\infty \quad \text{if } p<\bar{k} \text{ where } \beta_p=c_{im_i}/a_{im_i}. \end{aligned} \tag{3.25}$$

$$\bar{e}=b-\sum_{i\neq\bar{i}} a_{ij_i(\bar{k})}, \quad \text{where } \bar{i} \text{ is defined by } \beta_{\bar{k}}=c_{\bar{i}m_{\bar{i}}}/a_{\bar{i}m_{\bar{i}}} \text{ or}$$

$$\beta_{\bar{k}}=(c_{\bar{i}\bar{j}-1}-c_{\bar{i}\bar{j}})/(a_{\bar{i}\bar{j}-1}-a_{\bar{i}\bar{j}}),\ \bar{j}-1=j_{\bar{i}}(\bar{k}).$$

Theorem 3.1. Let

$$\begin{aligned} \bar{v}&=\sum_{i=1}^{n}\bar{\mu}_i(\bar{k})+b\bar{\lambda} \\ \bar{\mu}_i(\bar{k})&=c_{ij_i(\bar{k})}-a_{ij_i(\bar{k})}\bar{\lambda}, \quad i=1,2,\ldots,n, \end{aligned} \tag{3.26}$$

where $\bar{\lambda}$ and $j_i(\bar{k})$ are obtained in DUAL(P). Then $(\bar{\lambda}, \bar{\mu}_i(k), i=1,2,\ldots, n)$ is an optimal solution of $\bar{D}$ and $\bar{v}$ is its value. Furthermore DUAL(P) runs in $O(N \log N)$ time.

Proof: The optimality is an obvious concequence of the above argument and that $j_i(\bar{k})$ of (3.25) can be regarded as $j_i(\bar{\lambda})$ of (3.16). To show the time complexity, note that D5 is eventually reached since $v'<0$ holds for $k=1$ by (3.21). Now D1 requires $O(N \log N)$ time for sorting. The loop D2-D4 requires at most $O(N)$ time since this is essentially scanning N (or less) numbers β_N, $\beta_{N-1},\ldots, \beta_{\bar{k}}$ in this order. □

Although $j_i(\bar{k})$ of (3.25) does not contradict $j_i(\lambda)$ of (3.16) for $\lambda=\bar{\lambda}$, we note here that other choices may also be possible. (Note that (3.16) does not define $j_i(\lambda)$ uniquely.) In particular, if some β_k's take the same value, these may be numbered arbitrarily as far as (3.23) is satisfied. For each numbering, DUAL(P) may give a different set of $j_i(\bar{k})$'s. The optimal solution of $\bar{D}$ of Theorem 3.1 does not depend on such numberings, but the optimal solution of $\bar{P}$ constructed in Section 4 does depend. A modification of DUAL is therefore considered in Appendix to obtain a numbering which tends to yield an optimal solution of $\bar{P}$ more useful in the algorithm for solving P.

4. Construction of an Optimal Solution of $\bar{P}$

An optimal solution $\bar{x}$ of $\bar{P}$ is constructed from an optimal solution of $\bar{D}$ of Theorem 3.1. Consider the following two cases.

(i) $\beta_{\bar{k}}=c_{\bar{i}m_{\bar{i}}}/a_{\bar{i}m_{\bar{i}}}$ in DUAL(P): Then let $\bar{x}=(\bar{x}_{11}, \bar{x}_{12}, \ldots, \bar{x}_{nm_n})$ be given by

$$\begin{aligned} &\bar{x}_{ij_i(\bar{k})}=1, \quad \text{for } i\neq\bar{i} \text{ such that } j_i(\bar{k})<\infty \\ &\bar{x}_{\bar{i}m_{\bar{i}}}=\bar{e}/a_{\bar{i}m_{\bar{i}}} \quad (\bar{e} \text{ is given in (3.25)}) \\ &\bar{x}_{ij}=0 \quad \text{for other } i \text{ and } j. \end{aligned} \tag{4.1}$$

(ii) $\beta_{\bar{k}}=(c_{\bar{i}\bar{j}-1}-c_{\bar{i}\bar{j}})/(a_{\bar{i}\bar{j}-1}-a_{\bar{i}\bar{j}})$, where $\bar{j}-1=j_{\bar{i}}(\bar{k})$:

$$\begin{aligned} &\bar{x}_{ij_i(\bar{k})}=1, \quad \text{for } i\neq\bar{i} \text{ such that } j_i(\bar{k})<\infty \\ &\bar{x}_{\bar{i}\bar{j}}=\alpha, \ \bar{x}_{\bar{i}\bar{j}-1}=1-\alpha \\ &\bar{x}_{ij}=0 \quad \text{for other } i \text{ and } j, \end{aligned} \tag{4.2}$$

where

$$\alpha=(a_{\bar{i}\bar{j}-1}-\bar{e})/(a_{\bar{i}\bar{j}-1}-a_{\bar{i}\bar{j}}). \tag{4.3}$$

Theorem 4.1. $\bar{x}$ defined by (4.1) or (4.2) is an optimal solution of $\bar{P}$. $\bar{x}$ can be computed in $O(N)$ time from the optimal solution of $\bar{D}$ of Theorem 3.1.

Proof: By the duality theory of linear programming, it is sufficient to show that (4.1) or (4.2) is feasible in $\bar{P}$ and satisfies the complementary slackness condition with respect to the optimal solution of $\bar{D}$ of Theorem 3.1 (e.g., [3]).

(a) Feasibility. We first show that

$$0<\bar{e}\leq a_{\bar{i}m_{\bar{i}}} \text{ holds in (4.1), and}$$
$$a_{\bar{i}\bar{j}}<\bar{e}\leq a_{\bar{i}\bar{j}-1} \text{ holds in (4.2).} \tag{4.4}$$

In the first case, $0<\bar{e}$ follows since $v'>0$ and

$$v'=b-\sum_{i\neq\bar{i}} a_{ij_i(\bar{k})} \quad (=\bar{e} \text{ by (3.25)})$$

hold for $k=\bar{k}+1$ (note $j_{\bar{i}}(k)=\infty$). $\bar{e}\leq a_{\bar{i}m_{\bar{i}}}$ is also obvious since for $k=\bar{k}$

$$v'=b-\sum_{i=1}^{n} a_{ij_i(\bar{k})}$$
$$=b-\sum_{i\neq\bar{i}} a_{ij_i(\bar{k})}-a_{\bar{i}m_{\bar{i}}}=\bar{e}-a_{\bar{i}m_{\bar{i}}}\leq 0$$

hold in Step D4. The second relation of (4.4) can also be similarly proved. (4.4) then implies that $\bar{x}$ satisfies (3.3) (3.4) of $\bar{P}$. Next note that

$$a_{\bar{i}m_{\bar{i}}}\bar{x}_{\bar{i}m_{\bar{i}}}=\bar{e} \quad \text{in (4.1)}$$
$$a_{\bar{i}\bar{j}-1}\bar{x}_{\bar{i}\bar{j}-1}+a_{\bar{i}\bar{j}}\bar{x}_{\bar{i}\bar{j}}=a_{\bar{i}\bar{j}-1}(1-\alpha)+a_{\bar{i}\bar{j}}\alpha=\bar{e} \quad \text{(by (4.3)) in (4.2)}$$

hold. Thus

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i} a_{ij}\bar{x}_{ij}=\bar{e}+\sum_{i\neq\bar{i}} a_{ij_i(\bar{k})}=b \tag{4.5}$$

and (3.2) of $\bar{P}$ is satisfied. Therefore $\bar{x}$ is feasible in $\bar{P}$.

(b) Complementary slackness.

(b-1) For each positive dual variable λ or μ_i, it is shown that the corresponding primal condition is satisfied by equality. First $\bar{\lambda}>0$ holds and the corresponding constraint $\Sigma\Sigma a_{ij}x_{ij}\leq b$ is satisfied by equality as shown in (4.5). For $i\neq\bar{i}$, $\bar{\mu}_i(\bar{k})>0$ implies $j_i(\bar{k})\neq\infty$ (see (3.17)) and $\bar{x}_{ij_i(\bar{k})}=1$ by (4.1) or (4.2). Thus $\Sigma x_{ij}\leq 1$ is satisfied by equality. For $i=\bar{i}$,

$$\bar{\mu}_i(\bar{k})=c_{\bar{i}m_{\bar{i}}}-a_{\bar{i}m_{\bar{i}}}\bar{\lambda}=0$$

holds in case of (4.1) since $\bar{\lambda}=c_{\bar{i}m_{\bar{i}}}/a_{\bar{i}m_{\bar{i}}}$, and $\bar{\mu}_{\bar{i}}(\bar{k})>0$ holds in case of (4.2). In the latter case,

$$\sum_{j=1}^{m_i}\bar{x}_{\bar{i}j}=\bar{x}_{\bar{i}\bar{j}-1}+\bar{x}_{\bar{i}\bar{j}}=(1-\alpha)+\alpha=1$$

and $\Sigma x_{\bar{i}j}\leq 1$ is satisfied by equality.

(b-2) For each positive primal variable x_{ij}, it is shown that the corresponding dual constraint is satisfied by equality. For $i\neq\bar{i}$, $\bar{x}_{ij_i(\bar{k})}=1$ holds and

$$\mu_i=\bar{\mu}_i(\bar{k})=c_{ij_i(\bar{k})}-a_{ij_i(\bar{k})}\bar{\lambda}$$

holds by (3.26). For $i=\bar{i}$, $\bar{x}_{\bar{i}m_{\bar{i}}}>0$ holds in (4.1), and

$$0=\bar{\mu}_{\bar{i}}(\bar{k})=c_{\bar{i}m_{\bar{i}}}-a_{\bar{i}m_{\bar{i}}}\bar{\lambda}=0$$

Finally for $i=\bar{i}$ and (4.2), $\bar{x}_{\bar{i}\bar{j}-1}$ and $\bar{x}_{\bar{i}\bar{j}}$ are both positive. But

$$\begin{aligned}\bar{\mu}_{\bar{i}}(\bar{k})&=c_{\bar{i}\bar{j}-1}-a_{\bar{i}\bar{j}-1}\bar{\lambda}\quad(\text{by } j_{\bar{i}}(\bar{k})=\bar{j}-1)\\&=c_{\bar{i}\bar{j}-1}-a_{\bar{i}\bar{j}-1}(c_{\bar{i}\bar{j}-1}-c_{\bar{i}\bar{j}})/(a_{\bar{i}\bar{j}-1}-a_{\bar{i}\bar{j}})\\&=c_{\bar{i}\bar{j}}-a_{\bar{i}\bar{j}}\bar{\lambda}.\end{aligned}$$

Thus the corresponding constraints are satisfied by equality.

The result for the computational complexity is obvious. □

A similar optimal solution of $\bar{P}$ is also considered in [2], without using the concept of $\bar{D}$. The next theorem is an obvious consequence of the above construction.

Theorem 4.2. (a) The optimal solution $\bar{x}$ of $\bar{P}$ given by (4.1) or (4.2) either satisfies the multiple-choice constraint (1.4) or violates (1.4) for only one $i=\bar{i}$.

(b) $\bar{x}$ satisfies (1.4) if $\bar{x}$ is given by (4.1) (i.e., $\beta_{\bar{k}}=c_{\bar{i}m_{\bar{i}}}/a_{\bar{i}m_{\bar{i}}}$ holds in DUAL(P)) or if $\alpha=0$ holds in (4.2) (i.e., $\bar{e}=a_{\bar{i}\bar{j}-1}$).

(c) If $\bar{x}$ does not satisfy (1.4) for $i=\bar{i}$, only two successive variables $x_{\bar{i}\bar{j}-1}$ and $x_{\bar{i}\bar{j}}$ assume positive values among $j=1,2,\ldots,m_{\bar{i}}$. □

In other words, $\bar{x}$ is very close to an optimal solution of P, and hence it may be effectively used to solve P.

In view of Theorem 4.2 (b), it is desirable to have $\beta_{\bar{k}}=c_{\bar{i}m_{\bar{i}}}/a_{\bar{i}m_{\bar{i}}}$ or $\bar{e}=a_{\bar{i}\bar{j}-1}$ in DUAL(P) if possible. Since an arbitrary numbering is possible for these β_k's with the same value, as mentioned at the end of Section 3, a clever choice of a numbering may result in $\beta_{\bar{k}}=c_{\bar{i}m_{\bar{i}}}/a_{\bar{i}m_{\bar{i}}}$ or $\bar{e}=a_{\bar{i}\bar{j}-1}$. For this purpose, a modification of Algorithm DUAL is proposed in Appendix. The computational experiment reported in Sections 7 and 9 are done with this modified algorithm.

5. Approximate Solutions by Rounding

Let $\bar{x}$ be an optimal solution of $\bar{P}$ obtained in Section 4. If $\bar{x}$ satisfies the multiple-choice constraint (1.4), $\bar{x}$ is also an optimal solution of P. Therefore, assume that $\bar{x}$ does not satisfy (1.4), i.e.,

$$x_{\bar{i}\bar{j}-1},\ x_{\bar{i}\bar{j}} > 0, \tag{5.1}$$

where

$$\beta_{\bar{k}} = (c_{\bar{i}\bar{j}-1} - c_{\bar{i}\bar{j}})/(a_{\bar{i}\bar{j}-1} - a_{\bar{i}\bar{j}}),\ \bar{j}-1=j_{\bar{i}}(\bar{k}), \tag{5.2}$$

holds in DUAL(P) (see Theorem 4.2). (1.4) is satisfied for all i except $i=\bar{i}$.

From this $\bar{x}$, we construct the following solutions $x^{(1)}$ and $x^{(2)}$ feasible in P.

$$\begin{aligned} &x_{ij}^{(1)}=\bar{x}_{ij},\ \text{for } i\neq\bar{i},\ j=1,2,\ldots,m_i \\ &x_{\bar{i}\bar{j}}^{(1)}=1,\ x_{\bar{i}j}^{(1)}=0 \text{ for } j\neq\bar{j}. \end{aligned} \tag{5.3}$$

$$\begin{aligned} &x_{ij}^{(2)}=\bar{x}_{ij},\ \text{for } i\neq\bar{i},\ j=1,2,\ldots,m_i \\ &x_{\bar{i}\bar{j}-1}^{(2)}=\bar{e}/a_{\bar{i}\bar{j}-1},\ x_{\bar{i}j}^{(2)}=0 \text{ for } j\neq\bar{j}-1, \end{aligned} \tag{5.4}$$

where $\bar{e}$ is given by (3.25). Their objective values are respectively given by

$$z^{(1)}=\sum\sum c_{ij}x_{ij}^{(1)}=\sum_{i\neq\bar{i}} c_{ij_i(\bar{k})}+c_{\bar{i}\bar{j}} \tag{5.5}$$

$$z^{(2)}=\sum\sum c_{ij}x_{ij}^{(2)}=\sum_{i\neq\bar{i}} c_{ij_i(\bar{k})}+\bar{e}(c_{\bar{i}\bar{j}-1}/a_{\bar{i}\bar{j}-1}). \tag{5.6}$$

The better of $x^{(1)}$ and $x^{(2)}$ is called the *approximate solution by rounding* and denoted $x^{(R)}$. Its objective value is

$$z^{(R)}=\max[z^{(1)},\ z^{(2)}]. \tag{5.7}$$

Theorem 5.1. Assume that the optimal solution $\bar{x}$ of $\bar{P}$ does not satisfy (1.4) (i.e., $\bar{x}$ is given by (4.2)). Denote the optimal value of P by z^0. Then the approximate solution by rounding satisfies

$$z^{(R)}/z^0 > 1/2. \tag{5.8}$$

Proof: Since the LP optimal value $\bar{z}$ of $\bar{P}$ satisfies $\bar{z}\geq z^0$, we show

$z^{(R)}/\bar{z}>1/2$ instead of (5.8). By (4.2), $\bar{z}$ is given by

$$\bar{z}=\sum_{i\neq\bar{i}} c_{ij_i(k)}+c_{\bar{i}\bar{j}-1}(1-\alpha)+c_{\bar{i}\bar{j}}\alpha. \tag{5.9}$$

Next note that

$$\sum_{i\neq\bar{i}} c_{ij_i(k)}\geq 0,\ c_{\bar{i}\bar{j}}>c_{\bar{i}\bar{j}}\alpha \text{ and}$$

$$\bar{e}(c_{\bar{i}\bar{j}-1}/a_{\bar{i}\bar{j}-1})>c_{\bar{i}\bar{j}-1}(\bar{e}-a_{\bar{i}\bar{j}})/(a_{\bar{i}\bar{j}-1}-a_{\bar{i}\bar{j}})$$

$$=c_{\bar{i}\bar{j}-1}(1-\alpha).$$

These and (5.5) (5.6) (5.9) imply

$$z^{(1)}+z^{(2)}>\bar{z},$$

and hence $z^{(R)}=\max[z^{(1)}, z^{(2)}]>\bar{z}/2$. □

Theorem 5.2. The bound given in Theorem 5.1 for the approximate solution by rounding is sharp. In other words, there exists a multiple-choice knapsack problem P satisfying

$$z^{(R)}/z^{0}\leq(1/2)+\epsilon$$

for any $\epsilon>0$.

Proof: Consider the following problem P with $n=2$, $m_1=m_2=2$.

$$\begin{aligned} &c_{11}=1,\ c_{12}=\gamma,\ a_{11}=1,\ a_{12}=\delta \\ &c_{21}=1,\ c_{22}=1-2\gamma,\ a_{21}=2,\ a_{22}=1-\delta \\ &b=\gamma \end{aligned} \tag{5.10}$$

where γ and δ satisfy

$$1>>\gamma>>\delta>0. \tag{5.11}$$

For these coefficients, we have

$$\beta_1=\frac{c_{21}-c_{22}}{a_{21}-a_{22}}=\frac{2\gamma}{1+\delta}<\beta_2=\frac{c_{22}}{a_{22}}=\frac{1-2\gamma}{1-\delta}<\beta_3=\frac{c_{11}-c_{12}}{a_{11}-a_{12}}=\frac{1-\gamma}{1-\delta}$$

$$<\beta_4=\frac{c_{12}}{a_{12}}=\frac{\gamma}{\delta}.$$

Applying DUAL(P) on this set results in

$$\beta_{\bar{k}} = \beta_3 = \frac{1-\gamma}{1-\delta} \ (=\bar{\lambda})$$

$$\bar{i}=1,\ j_1(\bar{k})=1,\ j_2(\bar{k})=\infty \quad \text{(by (3.25))}$$

$$\bar{e}=b=\gamma,\ \alpha=(1-\gamma)/(1-\delta) \quad \text{(by (3.25) and (4.3))}.$$

$\bar{x}$ is given by (4.2) as follows

$$x_{11}=1-\alpha=(\gamma-\delta)/(1-\delta),\ x_{12}=\alpha=(1-\gamma)/(1-\delta)$$

(5.12) $$\bar{x}_{21}=\bar{x}_{22}=0$$

$$\bar{z}=(2\gamma-\delta-\gamma^2)/(1-\delta).$$

$x^{(1)}$ and $x^{(2)}$ are now given by (5.3) and (5.4).

$$x_{11}^{(1)}=0,\ x_{12}^{(1)}=1,\ x_{21}^{(1)}=x_{22}^{(1)}=0,\ z^{(1)}=\gamma$$

$$x_{11}^{(2)}=\gamma,\ x_{12}^{(2)}=x_{21}^{(2)}=x_{22}^{(2)}=0,\ z^{(2)}=\gamma.$$

Finally the optimal solution x^0 of P is given by

$$x_{11}^0=x_{21}^0=0,\ x_{12}^0=1,\ x_{22}^0=(\gamma-\delta)/(1-\delta)$$

$$z^0=(2\gamma+2\gamma\delta-2\gamma^2-\delta)/(1-\delta)$$

(the optimality is checked by exhausting all cases). Thus

$$z^{(R)}/z^0=(\gamma-\gamma\delta)/(2\gamma+2\gamma\delta-2\gamma^2-\delta)$$

$$\xrightarrow[\gamma \to 0,\ \ \delta/\gamma \to 0]{} \frac{1}{2},$$

proving the theorem statement. □

Note however that the theoretical bound 1/2 is attainable only if $\sum_{i\neq\bar{i}} c_{ij_i(\bar{k})}=0$ holds in $\bar{x}$. In most cases, especially when n and b are large, $\sum_{i\neq\bar{i}} c_{ij_i(\bar{k})}$ takes rather large value and $z^{(R)}/z^0$ becomes closer to 1. This tendency is actually confirmed in the computational experiment in Section 7.

Finally we mention the computational complexity for obtaining $x^{(R)}$. Since (5.3) and (5.4) can be computed in $O(N)$ time from $\bar{x}$, the entire process to compute $x^{(R)}$ from a given P requires $O(N \log N)$ time. The sorting time in Step D1 of DUAL(P) is dominant in determining the total computation time.

6. Approximate Solutions by Breadth-1 Search

A higher order approximate algorithm called breadth-K search is investigated in [9], as a generalization of the approximate solution by rounding. For a given $\varepsilon>0$, the selection of an appropriate K can yeiled an approximate solution $x^{(B)}$ with its objective value $z^{(B)}$ satisfying

$$z^{(B)}/z^{0}>1-\varepsilon,$$

in computation time bounded by a polynomial of N. In this section, only an outline of the approximate algorithm by breadth-1 (i.e., K=1) search is given. Its computational results are also included in Section 7.

First we introduce the third approximate solution $x^{(3)}$ of P, in addition to $x^{(1)}$ and $x^{(2)}$. Assume that

$$\beta_{\bar{k}}=\bar{\lambda}=(c_{\bar{i}\bar{j}-1}-c_{\bar{i}\bar{j}})/(a_{\bar{i}\bar{j}-1}-a_{\bar{i}\bar{j}}) \tag{6.1}$$

holds in DUAL(P). Although $c_{\bar{i}\bar{j}-1}/a_{\bar{i}\bar{j}-1}$ is not stored in list L of DUAL(P), it is inserted into L and β's in L are renumbered so that (3.23) is preserved. Let

$$\begin{aligned}&\beta_{k'}=\lambda'=c_{\bar{i}\bar{j}-1}/a_{\bar{i}\bar{j}-1} \quad (>\bar{\lambda})\\ &A'=\sum_{i\neq\bar{i}} a_{ij_i(k')} \quad (<b),\end{aligned} \tag{6.2}$$

where $j_i(k')$ is given by (3.25) with $\bar{k}$ replaced by k'. Then $x^{(3)}$ is obtained as follows:

(i) If $A'+a_{\bar{i}\bar{j}-1}\geq b$, then

$$\begin{aligned}&x^{(3)}_{ij_i(k')}=1 \text{ for } i\neq\bar{i},\ x^{(3)}_{ij}=0 \text{ for } i\neq\bar{i},\ j\neq j_i(k')\\ &x^{(3)}_{\bar{i}\bar{j}-1}=(b-A')/a_{\bar{i}\bar{j}-1},\ x^{(3)}_{\bar{i}\bar{j}}=0 \text{ for } j\neq\bar{j}-1\\ &z^{(3)}=\sum_{i\neq\bar{i}} c_{ij_i(k')}+c_{\bar{i}\bar{j}-1}(b-A')/a_{\bar{i}\bar{j}-1}.\end{aligned} \tag{6.3}$$

(ii) If $A'+a_{\bar{i}\bar{j}-1}<b$, then $x^{(3)}$ is not computed and

$$z^{(3)}=-\infty \tag{6.4}$$

$x^{(3)}$ obtained in (6.3) is obviously feasible in P. It is the LP optimal solution of $\bar{P}$ with all $x_{\bar{i}j}$, $j\neq\bar{j}-1$, fixed to 0. (The condition $A'+a_{\bar{i}\bar{j}-1}\geq b$ guarantees that the LP optimal solution satisfies condition (1.4).)

Finally, let the best of $x^{(1)}$, $x^{(2)}$ and $x^{(3)}$ be denoted $x^{(T)}$ and let

$$z^{(T)}=\max[z^{(1)}, z^{(2)}, z^{(3)}]. \tag{6.5}$$

In order to compute $x^{(3)}$, it is necessary to have λ' and A' of (6.2) calculated for all c_{ij-1}/a_{ij-1} (note that c_{ij-1}/a_{ij-1} are not stored in L for P). This can be done in $O(N)$ steps, however, by storing all c_{ij-1}/a_{ij-1} implicitly together with $\beta_1, \beta_2, \ldots, \beta_N$ of (3.23) when list L is prepared in DUAL(P). The detail is omitted.

The approximate algorithm by breadth-1 search first obtains the LP optimal solution $\bar{x}$ of $\bar{P}$. If $\bar{x}$ is feasible in P, P is solved. Otherwise $x^{(T)}$ is calculated and a new problem P ($x_{\bar{i}\bar{j}-1}=0$) is generated, i.e., $x_{\bar{i}\bar{j}-1}$ is fixed to 0 in P. The same procedure is then repeated for P ($x_{\bar{i}\bar{j}-1}=0$) until the LP optimal solution of a tested problem Q becomes feasible in Q. This process eventually terminates since the LP optimal solution of Q is trivially feasible in Q if all N variables x_{ij} are fixed to 0. The best feasible solution obtained in this process is the approximate solution obtained by breadth-1 search. It is denoted by $x^{(B)}$ and its value by $z^{(B)}$.

Algorithm APPRXB1(P)

B1 : $Q\leftarrow P$, $z\leftarrow -\infty$

B2 : Obtain the LP optimal solution $\bar{x}$ and its value $\bar{z}$ of Q (i.e., DUAL(Q) is executed). If $\bar{x}$ is feasible in Q, then

$$z^{(B)}\leftarrow\max[z, \bar{z}]$$

and halt. Otherwise, calculate $x^{(T)}$ and its value $z^{(T)}$ of Q and let

$$z\leftarrow\max[z, z^{(T)}].$$

B3 : $Q\leftarrow Q(x_{\bar{i}\bar{j}-1}=0)$, where $\beta_{\bar{k}}=(c_{\bar{i}\bar{j}-1}-c_{\bar{i}\bar{j}})/(a_{\bar{i}\bar{j}-1}-a_{\bar{i}\bar{j}})$ holds in DUAL(Q). Return to B2. □

As discussed so far, the first Q (i.e., P) requires $O(N \log N)$ time in B2. If we treat other problems in the same manner, the total time is $O(N^2 \log N)$ since at most N problems are generated. However, $Q(x_{\bar{i}\bar{j}-1}=0)$ can be more efficiently treated by judiciously using the data and results obtained for Q. Denote $\bar{k}$, $\bar{\lambda}$, v', j_i, $\bar{e}$ obtained in DUAL(Q) by $\bar{k}(Q)$, $\bar{\lambda}(Q)$, $v'(Q)$, $j_i(Q)$, $\bar{e}(Q)$. When $x_{\bar{i}\bar{j}-1}=0$ is imposed to Q,

$$\beta_{\bar{k}(Q)}=\frac{c_{\bar{i}\bar{j}-1}-c_{\bar{i}\bar{j}}}{a_{\bar{i}\bar{j}-1}-a_{\bar{i}\bar{j}}}, \quad \beta_{k'}=\frac{c_{\bar{i}\bar{j}-2}-c_{\bar{i}\bar{j}-1}}{a_{\bar{i}\bar{j}-2}-a_{\bar{i}\bar{j}-1}}$$

($\beta_{k'}$ does not exist if $\bar{j}-1=1$) are deleted from $L=\{\beta_k\}$, and then

$$\beta_\ell = \frac{c_{\bar{i}\bar{j}}-2^{-c_{\bar{i}\bar{j}}}}{a_{\bar{i}\bar{j}}-2^{-a_{\bar{i}\bar{j}}}}$$

is usually added (if $\bar{j}-1=1$, none is added). It is easy to show

$$\beta_{k'} < \beta_\ell < \beta_{\bar{k}(Q)} \tag{6.6}$$

and hence β_k's located to the right of $\beta_{\bar{k}(Q)}$ in L do not change by this modification. The search for $\bar{k}$ of $Q(x_{\bar{i}\bar{j}-1}=0)$ is then resumed from $k=\bar{k}(Q)+1$ (i.e., the one located next to the right of $\beta_{\bar{k}(Q)}$) to its left on the modified list. Namely, letting

$$\begin{aligned} &k \leftarrow \bar{k}(Q)+1 \\ &v' \leftarrow v'(Q)+(a_{\bar{i}\bar{j}-1}-a_{\bar{i}\bar{j}}) \\ &j_{\bar{i}}(k) \leftarrow \bar{j},\ j_i(k) \leftarrow j_i(Q) \text{ for } i \neq \bar{i}, \end{aligned} \tag{6.7}$$

Algorithm DUAL is entered from Step D4 ($v'>0$ holds at the first entrance), and executed until D5 is reached. The result gives an LP optimal solution of $\bar{Q}(x_{\bar{i}\bar{j}-1}=0)$. The computation time required for this process should be much less than that required for solving $Q(x_{\bar{i}\bar{j}-1}=0)$ from scratch.

Strictly speaking, there may arise some complication in the above process, if some variables suppressed from list L of Q due to (3.15) have revived as a result of deleting variable $x_{\bar{i}\bar{j}-1}$ from Q. Some adjustment of L is then necessary. However basically the same procedure is still applicable, and the running time of APPRXB1 is at most $O(mN \log N)$ for sorting and $O(mN)$ for the rest, as discussed in [9], where $m=\max_i m_i$.

We conclude this section by giving the next theorem proved in [9].

Theorem 6.1. $z^{(B)}$ obtained in APPRXB1(P) satisfies

$$z^{(B)}/z^0 > \frac{3}{4},$$

and furthermore this bound is sharp. □

7. Computational Experiments of Approximate Algorithms

The approximate algorithms of Sections 5 and 6 are coded† in FORTRAN,

† Approximate solution $x^{(3)}$ of (6.3) is not incorporated to define $z^{(B)}$ of APPRXB1(P) in our implementation.

and run on FACOM 230/60 and M190 of Data Processing Center of Kyoto University. QUICKSORT (e.g., [1] [13]) is used as the sorting algorithm in Step D1 of DUAL. FACOM 230/60 and M190 are roughly equivalent to IBM 360/60 and 370/175, respectively. Although the accurate comparison is difficult, M190 seems to be 10~15 times faster than 230/60.

Three types of problems, A, B, C, are randomly generated for test. For simplicity, $m_i=2$ is assumed for $i=1,2,\ldots, n$ in all problems.

A. Coefficients a_{i1}, a_{i2}, c_{i1}, c_{i2}, $i=1,2,\ldots, n$, are integers randomly taken from the uniform distribution with range [1, 1000]. b is then determined by

$$b=d\sum_{i=1}^{n}(a_{i1}+a_{i2}), \tag{7.1}$$

where d is a parameter specifying the tightness of constraint (1.2). A small d implies a tight constraint. Before applying approximate algorithms, unnecessary variables are deleted according to the properties discussed in (2.2) and (2.3). After this, only about $n/4$ i's have both x_{i1} and x_{i2} in P (as easily calculated since coefficients a, c are generated independently). If the resulting problem does not satisfy (2.4), it can be trivially solved. Note that the assumption $m_i=2$ ($m_i\leq 2$ after the simplifications (2.2) and (2.3)) helps to simplify the implementation since only one β_k of type $(c_{ij-1}-c_{ij})/(a_{ij-1}-a_{ij})$ exists for each i and the simplification based on (3.14) is not necessary.

B. This type is generated to see the performance of approximate algorithms when many β_k's in (3.23) take the same values. Coefficients a_{i2} and c_{i2} are generated as in type A, but a_{i1} and c_{i1} are determined by

$$a_{i1}=a_{i2}+100,\quad c_{i1}=c_{i2}+100,\quad i=1,2,\ldots, n. \tag{7.2}$$

In this type of problems, (2.2) is automatically satisfied and all $(c_{i1}-c_{i2})/(a_{i1}-a_{i2})$ take the same value 1 for $i=1,2,\ldots, n$. Coefficient b is determined by (7.1).

C. As a model of difficult problems, coefficients of type C problems are generated as follows.

$$\begin{aligned} &a_{i1}=200,\quad a_{i2}=100\\ &c_{i2}=\max[1000+100\delta_i,\ 0] \qquad (7.3)\\ &c_{i1}=c_{i2}+100\cdot\max[\delta_i,\ 0],\quad i=1,2,\ldots, n, \end{aligned}$$

where δ_i are random numbers taken from the normal distribution $N(t, \sigma^2)$ with

(7.4) $t=20.0,\ \sigma=5.0,\ 10.0.$

If we neglect the truncation effects caused by max[] in (7.3), parameters β_k are given by

(7.5) $$(c_{i1}-c_{i2})/(a_{i1}-a_{i2})=\delta_i$$
$$c_{i1}/a_{i1}=5+\delta_i,\ c_{i2}/a_{i2}=10+\delta_i,$$

which are normally distributed around means 20.0, 25.0 and 30.0 respectively. (2.2) and (2.3) are always satisfied for these coefficients. Coefficient b is set to

(7.6) $$b=\frac{n}{2}(a_{i1}+a_{i2})+0.7(a_{i1}-a_{i2})=150n+70$$

for the following reason. If the numbers of δ_i's greater than and smaller than its mean $t=20.0$ are about the same, b of (7.6) makes $\bar{\lambda}=\beta_{\bar{k}}$(obtained by DUAL($P$)) close to $t=20.0$, because $\bar{x}_{i1}=1$ holds for about $n/2$ i's and $\bar{x}_{i2}=1$ for other i's (except possibly one i for which $\bar{x}_{i1},\ \bar{x}_{i2}>0$) in the LP optimal solution $\bar{x}$. Thus most of β_k's around $\beta_{\bar{k}}$ (including $\beta_{\bar{k}}$) are of $(c_{i1}-c_{i2})/(a_{i1}-a_{i2})$ type, and it tends to make $\bar{x}$ relatively far from the optimal solution of P. The constant 70 in (7.6) is introduced as a perturbation since otherwise the case of Theorem 4.2 (b) (the case of $\bar{e}=a_{\bar{i}\bar{j}-1}$) always occurs.

The computational results for type A problems are summarized in Tables 7.1~7.3. These were run on FACOM 230/60. 10 problems were generated for each n=100, 200,..., 1000. Table 7.1 shows the number of problems (out of 10) for which either (2.4) is not satisfied (i.e., trivially solved) or LP optimal solutions $\bar{x}$ are feasible in P (i.e., optimal in P). For a large d, the number increases because (2.4) tends to be not satisfied. For a small d, it also increases because $\bar{x}$ tends to solve P because $\bar{\lambda}$ is large (and case (b) of Theorem 4.2 is likely to occur). We see that the approximate algorithm by rounding is successful to obtain exact optimal solutions for the majority of type A problems.

To show the accuracy of approximate solutions, Table 7.2 lists the maximum of

(7.7) $$((\bar{z}-z^{(*)})/\bar{z})\times 100,\ \text{where } *=R,\ B,$$

for type A problems (all cases d=0.2, 0.3,..., 0.6 are considered for each n). Note that the ratio is computed against $\bar{z}$ (not z^0). Thus the real accuracy

Table 7.1. The number of type A problems (out of 10) for which the LP optimal solutions $\bar{x}$ is feasible (i.e., optimal) in P or (2.4) is not satisfied (i.e., trivially solved).

d \ n	100	200	300	400	500	600	700	800	900	1000
0.2	10	10	9	7	9	10	9	8	10	10
0.3	9	7	8	8	8	10	8	10	9	9
0.4	5	9	7	8	6	6	6	5	5	4
0.5	5	8	3	5	4	7	8	6	4	7
0.6	10	10	10	10	10	10	10	10	10	10

Table 7.2. The maximum percentage deviation of the approximate value $z^{(*)}$ from the LP optimal value $\bar{z}$, $\max((\bar{z}-z^{(*)})/\bar{z})\times 100$, $*=R$, B, among 50 type A problems for each n.

n	100	200	300	400	500	600	700	800	900	1000
Ronding $z^{(R)}$	0.10	0.07	0.04	0.03	0.03	0.03	0.02	0.01	0.03	0.02
Breadth-1 $z^{(B)}$	0.04	0.03	0.01	0.01	0.01	0.02	0.002	0.01	0.001	0.001

with respect to z^0 should be higher than that of Table 7.2. The computation time is given in Table 7.3. We see that DUAL(P) consumes a dominant part of total computation time, which is essentially the sorting time of $\beta_1, \beta_2, \ldots, \beta_N$. The total computation time seems to be $O(N \log N)$ ($=O(n \log n)$ since $m_i \leq 2$). From Tables 7.1-7.3, we may conclude that approximate algorithms are efficient enough to process very large problems, and yet its accuracy is extremely high. It is also interesting to note that the accuracy becomes even higher for problems of larger sizes. Breadth-1 search may be worthwhile since

Table 7.3. Average computation time for type A problems, in milli-seconds on FACOM 230/60. (Only problems satisfying (2.4) are considered in this table. Total includes some CPU time required for printing out. "Breadth-1" lists the average computation time for the problems in which LP optimal solutions $\bar{x}$ does not solve P.)

n	100	200	300	400	500	600	700	800	900	1000
DUAL	59	120	195	272	341	426	485	554	629	732
Rounding	1	1	1	2	1	1	1	1	1	2
Breadth-1	2	4	3	2	5	3	4	2	3	4
Total	69	140	221	302	386	484	543	621	700	814

it also runs very quickly and gains a noticeable improvement.

Type B problems are then solved in a similar manner, 10 problems each for n=100, 200,..., 1000 and d=0.2, 0.3,..., 0.6. The computation time is about the same as type A problems (and hence is not given). A significant difference from type A is that LP optimal solutions $\bar{x}$ are always feasible in P. This is because the modified DUAL of Appendix is quite powerful in finding a numbering of $\beta_k=(c_{i1}-c_{i2})/(a_{i1}-a_{i2})$ (=1 for all i) which results in $\bar{e}=a_{\overline{ij}-1}$ (the case of Theorem 4.2 (b)). Although the structure of type B problems may be too restricted, we may conclude that approximate algorithms tend to be more accurate if many β_k's take the same values.

Finally, computational results for type C problems are summarized in Tables 7.4 and 7.5. Only approximate solutions by rounding are computed for these problems. Comparing Table 7.5 with Table 7.2, we see that the accuracy of the obtained approximate solutions is about the same as that of type A problems. However, the number of type C problems for which exact optimal solutions are obtained by the approximate method (Table 7.4) is noticeably less than the case of type A problems (Table 7.1). This may indicate that type C problems are more difficult than type A problems, in order to obtain exact optimal solutions. (This tendency is actually confirmed in the experiment of Section 9.) The computation time for type C problems is about the same as type A problems, and the detailed statistics are not cited.

Table 7.4. The number of type C problems (out of 10) for which the LP optimal solution $\bar{x}$ is feasible (i.e., optimal) in P or (2.4) is not satisfied (i.e., trivially solved).

d \ n	20	40	60	80	100	120	140
5.0	1	1	1	0	3	2	0
10.0	6	4	7	2	4	6	2

Table 7.5. The maximum percentage deviation of the approximate value $z^{(R)}$ by rounding from the LP optimal value $\bar{z}$, $\max((\bar{z}-z^{(R)})/\bar{z})\times 100$, among 10 type C problems for each n and σ.

σ \ n	20	40	60	80	100	120	140
5.0	0.44	0.21	0.14	0.11	0.08	0.07	0.06
10.0	0.41	0.20	0.13	0.10	0.08	0.07	0.06

For further details of computational results of approximate algorithms, see Teranaka [17].

8. Branch-and-Bound Algorithm for Exact Optimal Solutions

Since the multiple-choice knapsack problem is NP-complete as mentioned in Section 1, it is unlikely that there exists an exact algorithm which always runs in time bounded by a polynomial of N and $\log b$. This does not deny, however, that an exact algorithm solves most problems efficiently and the average time is of polynomial order. To explore this possibility, a branch-and-bound algorithm is proposed in this section. As reported in the next section, its average computation time is roughly $O(N \log N)$ in certain cases. Similar branch-and-bound algorithms are also discussed in [16] for the discrete version of the multiple-choice knapsack problem.

The branch-and-bound algorithm of this section is constructed by following the standard recipe described in papers such as [15] [14], in particular [7] [8]. Therefore we describe only basic ideas necessary to specify a branch-and-bound algorithm.

(1) Branching operation. Each partial problem P_t (including the original problem $P=P_0$) generated in the algorithm is also a multiple-choice knapsack problem. If P_t is not terminated by the test applied to P_t, it is decomposed into the following two partial problems:

$$P_{t1}=P_t(x_{\bar{i}\bar{j}-1}=0),\ P_{t2}=P_t(x_{\bar{i}\bar{j}}=0), \tag{8.1}$$

where

$$\beta_{\bar{k}}=(c_{\bar{i}\bar{j}-1}-c_{\bar{i}\bar{j}})/(a_{\bar{i}\bar{j}-1}-a_{\bar{i}\bar{j}}) \tag{8.2}$$

holds in DUAL(P_t). A partial problem is therefore specified by a set of variables x_{ij} fixed to 0.

(2) Upper bound of the optimal value of $\bar{P}_t$. The LP optimal value of $\bar{P}_t$ is used for this purpose. The LP relaxation $\bar{P}_t$ is also used as a test to terminate P_t; P_t is terminated if its LP optimal solution $\bar{x}$ solves P_t. As seen from the computational results in Section 7, this termination occurs very frequently.

(3) Lower bound of the optimal value of P_t. The objective values of approximate solutions of P_t are used. Although $z^{(R)}$ of (5.7) is used in our implementation, $z^{(B)}$ of Section 6 can also be used (probably more desirable).

(4) Search strategy. Search strategy determines the order in which generated partial problems P_t are tested in a branch-and-bound algorithm. Although any search strategy can be used in principle, depth-m search [6] based on the LP optimal value is coded in our implementation in order to test a variety of search strategies relalizable by changing parameter m. According to our computational results, depth-first search is best from the view point of total computation time (though it sometimes generates more partial problems than other search strategies such as best-bound search (e.g., [7] for the theoretical results)).

It is now possible in principle to implement a branch-and-bound algorithm from the above constituents. It is however important from the view point of efficiency to consider how data of each partial problem P_t are maintained and updated. This aspect is briefly sketched below.

In our implementation, lists L storing $\beta_1, \beta_2, \ldots, \beta_N$ are not prepared for all nodes, but only one master list is maintained. Each P_t has the data

necessary to modify the master list into the list for P_t, assuming that the current partial problem $P_{t'}$, represented by the master list is known. Given $P_{t'}$, it is straightforward to obtain the list for P_t if P_t is directly generated from $P_{t'}$, or $P_{t'}$ is directly generated from P_t. If P_t and $P_{t'}$ are distant, however, we first obtain the list for the common (closest) ancestor $P_{t''}$ of P_t and $P_{t'}$, by following reversely the sequence of decompositions from $P_{t'}$ to $P_{t''}$, and then modify it for P_t by following the sequence of decompositions from $P_{t''}$ to P_t. This explains why depth-first search outperforms other search strategies in our experiment. (P_t tested next to $P_{t'}$, tends to be in a position close to $P_{t'}$, if depth-first search is used.)

Next consider how $\bar{P}_{t1}$ and $\bar{P}_{t2}$ of (8.1) are solved by making use of the result for $\bar{P}_t$. Denote $\bar{k}$ and v' obtained in DUAL (P_t) by $\bar{k}(P_t)$ and $v'(P_t)$. P_{t1} is then treated in the same manner as $Q(x_{\bar{i}\bar{j}-1}=0)$ discussed at the end of Section 6. $P_{t2}=P_t(x_{\bar{i}\bar{j}}=0)$ can be similarly treated, but the search for $\bar{k}(P_{t2})$ is done in the reverse direction on list L. To see this, let

$$\beta_{\bar{k}(P_t)}=\frac{c_{\bar{i}\bar{j}-1}-c_{\bar{i}\bar{j}}}{a_{\bar{i}\bar{j}-1}-a_{\bar{i}\bar{j}}} \quad \text{and} \quad \beta_{k'}=\frac{c_{\bar{i}\bar{j}}-c_{\bar{i}\bar{j}+1}}{a_{\bar{i}\bar{j}}-a_{\bar{i}\bar{j}+1}} \quad \left(\text{or } \frac{c_{\bar{i}\bar{j}}}{a_{\bar{i}\bar{j}}} \text{ if } \bar{j}=m_{\bar{i}}\right)$$

be deleted from list L for P_t, and let

$$\beta_{\ell}=\frac{c_{\bar{i}\bar{j}-1}-c_{\bar{i}\bar{j}+1}}{a_{\bar{i}\bar{j}-1}-a_{\bar{i}\bar{j}+1}} \quad \left(\text{or } \frac{c_{\bar{i}\bar{j}-1}}{a_{\bar{i}\bar{j}-1}} \text{ if } \bar{j}=m_{\bar{i}}\right)$$

be added. (If some other variables which have been suppressed by (3.15) revive as a result of deleting $x_{\bar{i}\bar{j}}$, this process becomes more complicated, but can be similarly done [9].) It is easy to see

$$\beta_{\bar{k}(P_t)}<\beta_{\ell}<\beta_{k'}. \tag{8.3}$$

Also $v'=v'(P_t)\leq 0$ holds at the end of DUAL(P_t). After deleting $\beta_{\bar{k}}(P_t)$ and $\beta_{k'}$ from L, and inserting β_{ℓ}, k is increased from $k=\bar{k}(P_t)$ (i.e., the search is directed to the right of $\beta_{\bar{k}(P_t)}$) until $v'\geq 0$ is reached (see (3.22)). At this point, the computation halts and an LP optimal solution of $\bar{P}_t$ is constructed.

Further details of the above algorithm are given in [10].

9. Computational Experiments of Branch-and-Bound Algorithm

The branch-and-bound algorithm explained in the previous section is coded in FORTRAN and type A, C problems (defined in Section 7) are solved on

Table 9.1. Computational results of the branch-and-bound algorithm for type A problems.
(The average of 50 problems for each n and d.)

d \ n	50		100		200		300		400		500	
	Number[(a)]	Time[(b)]	Number	Time	Number	Time	Number	Time	Number	Time	Number	Time
0.2	1.4	34	1.2	44	1.3	63	1.2	96	1.3	110	1.3	121
0.3	2.0	36	1.5	44	1.7	64	1.3	97	1.6	111	1.5	122
0.4	1.9	35	2.2	45	2.1	65	2.1	100	2.3	121	2.0	129
0.5	3.6	24	7.8	37	4.5	54	5.4	88	5.3	108	9.0	135
0.6	1.0	7	1.0	14	1.0	27	1.0	55	1.0	62	1.0	66

(a) Number: The number of partial problems (including P_0) generated prior to termination.

(b) Time: Total computation time in milliseconds on FACOM M190.

Table 9.2. Analysis of computation time required for type A problems with n=500. (The average of 50 problems.)

d	Phase I[(c)]	Phase II[(d)]	Phase III[(e)]	Total
0.2	119	0	2	121
0.3	118	0	4	122
0.4	117	1	11	129
0.5	90	20	25	135
0.6	66	0	0	66

(c) Phase I : Time in milliseconds required for the initial setting of parameters, sorting parameters β_k, solving $\bar{P}_0$ (i.e., DUAL(P_0)), and obtaining the approximate solution of P_0 by rounding.

(d) Phase II : Time in milliseconds required for solving all $\bar{P}_t$'s and obtaining approximate solutions of P_t's, where P_t's are partial problems (excluding P_0) generated during the branch-and-bound computation.

(e) Phase III : Time in milliseconds required for constructing the computed optimal solution of P_0. Phase III is not necessary if only P_0 is generated as a partial problem.

FACOM M190. Depth-first search is exclusively used unless otherwise stated. The sorting algorithm implemented in this code is MERGE SORT (e.g., [1] [13]).

Table 9.1 summarizes the results for type A problems. Each figure is the average of 50 problems (which are different from those used in Section 7). As shown in Table 7.1, exact optimal solutions of P_0 (i.e., the original problem P) are usually obtained as approximate solutions of P_0 for the majority of

type A problems. In such cases, the branch-and-bound algorithm of course terminates after generating only P_0. i.e., the number of generated partial problems is 1. In other cases, P_0 is decomposed into finer and finer partial problems. Table 9.1 however shows that the numbers of the generated partial problems are relatively small in all cases. Interestingly enough, the number is almost independent of the problem size n, though it is dependent on parameter d (defined in (7.1)). Table 9.2 analyzes the computation time consumed in each phase of the algorithm, for type A problems with n=500. It shows that phase II consumes only a small fraction of the total time, indicating that the data structure and its management explained in Section 8 is quite successful to reduce the computation time. Since phase I is highly dominant in computation time, we may conclude that the average time required to solve type A problems is roughly equal to the time required to solve P_0 by an approximate algorithm, which is $O(N \log N)$.

Table 9.3. Computational results of the branch-and-bound algorithm for type C problems. (The average of 10 problems for each n and σ).

σ	n	20	40	60	80	100	120	140
5.0	Number[(a)]	9	135	309	404	580	786	2107
	Time[(b)]	167	198	443	606	900	1506	4096
10.0	Number[(a)]	4	5	6	7	8	3	6
	Time[(b)]	36	45	49	67	73	71	106

(a)(b) : See footnotes to Table 9.1.

Table 9.3 lists the results for type C problems. Contrary to type A problems, the number of the generated partial problems and the computation time seem to grow exponentially with n, when σ=5.0 is used. As mentioned in Section 1, the NP-completeness of the multiple-choice knapsack problem strong-

ly suggests the existence of instances of difficult problems. Our computational results suggest that type C problems with σ=5.0 may be such difficult instances. Note however that the difficulty of type C problems is highly sensitive to parameter σ. For σ=10.0, problems become much easier, and the required time seems to be $O(N \log N)$.

Finally, to see the effects of search strategies in the branch-and-bound algorithm, type C problems with σ=5.0 (which are different from those used in Table 9.3) are solved with both depth-first search and best-bound search.

Table 9.4. Comparison of search strategies for type C problems with σ=5.0. (The average of 10 problems for n=80, 100, and 7 problems for n=120.)

n	80		100		120(f)	
	Number(a)	Time(b)	Number	Time	Number	Time
Best-Bound	206	1897	187	1521	95	709
Depth-First	245	388	219	254	100	257

(a)(b) : See footnotes to Fig. 9.1.

(f) : Three problems could not be solved because the memory space bound (1000 partial problems) was exceeded.

Judging from Table 9.4, best-bound search (which is theoretically known to minimize the number of the generated partial problems) generates slightly less number of partial problems than depth-first search. Depth-first search, however, is definitely better from the view point of computation time. The reason for this was discussed in Section 8. The numbers of problems for which best-bound search generates less number of partial problems than depth-first search are as follows.

3 problems out of 10 for n=80

2 problems out of 10 for n=100

3 problems out of 7 for n=120 (other 3 problems could not be solved because the memory space bound was exceeded).

However, except for only two problems with n=120, depth-first search always requires less computation time than best-bound search.

Iwase [10] contains some further details of the computational results of the branch-and-bound algorithm.

10. Conclusion

Two approximate algorithms and one exact branch-and-bound algorithm are proposed for the multiple-choice knapsack problem. Judging from our computational experience, approximate algorithms are quite fast and yet give extremely good approximate solutions. The exact algorithm is also quite efficient and can be practical except for some difficult problems deliberately constructed. We may thus conclude that the multiple-choice knapsack problem is a rather easy one in the sense of the average computation time among many NP-complete problems. There are other NP-complete problems considered tractable in the sense of the average time, e.g., the ordinary knapsack problem, the set covering problem, the set packing problem and a certain class of scheduling problems. It would be necessary to investigate what makes these problems easier than other NP-complete problems. [4] contains such an attempt, and classifies NP-complete problems into two classes; those which are NP-complete in the strong sense and those which are not. The multiple-choice knapsack problem is not NP-complete in the strong sense, and this may give a reason why it is rather tractable among other NP-complete problems.

The ideas used in this paper may be extended to more general problems. For example, the LP optimal solution can be obtained in a similar manner even if constraint (1.4) is generalized as follows.

At most k_i of $x_{i1}, x_{i2}, \ldots, x_{im_i}$ are positive for i=1, 2,..., n, where k_i is a positive integer satisfying $1 \leq k_i \leq m_i$.

Therefore, approximate and exact algorithms may also be constructed in a similar manner. These extensions may be subjects of the future research.

References

[1] Aho, A.V., Hopcroft, J.E., and Ullman, J.D.: The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, Mass., 1974.

[2] Chandra, A.K., Hirschberg, D.S., and Wong, C.K.: Approximate Algorithms for the Knapsack Problem and Its Generalizations. IBM Research Report RC5616, IBM T. J. Watson Research Center, 1975.

[3] Dantzig, G.B.: Linear Programming and Extensions, Princeton University Press, Princeton, 1963.

[4] Garey, M.R. and Johnson, D.S.: "Strong" NP-Complete Results: Motivation, Examples and Implications. Submitted to Journal of the ACM, 1976.

[5] Ibaraki, T.: Complementary Programming. *Operations Research*, Vol. 19 (1971), pp. 1523-1528.

[6] Ibaraki, T.: Depth-*m* Search in Branch-and-Bound Algorithms. Working Paper, Department of Applied Mathematics and Physics, Kyoto University, 1974; to appear in *International Journal of Computer and Information Sciences*.

[7] Ibaraki, T.: Theoretical Comparisons of Search Strategies in Branch-and-Bound Algorithms. *International Journal of Computer and Information Sciences*, Vol. 5 (1976), pp. 315-344.

[8] Ibaraki, T.: The Power of Upper and Lower Bounding Functions in Branch-and-Bound Algorithms. Working Paper, Department of Applied Mathematics and Physics, Kyoto University, 1976.

[9] Ibaraki, T.: Approximate Algorithms for the Multiple-Choice Continuous Knapsack Problem. In preparation.

[10] Iwase, J.: A Branch-and-Bound Algorithm for the Complementary Knapsack Problem. Master Thesis, Department of Applied Mathematics and Physics, Kyoto University, 1977.

[11] Johnson, D.S., Demers, A., Ullman, J.D., Garey, M.R., and Graham, R.L.: Worst Case Performance Bounds for Simple One-Dimensional Packing Algorithms. *SIAM Journal on Computing*, Vol. 3 (1974), pp. 299-325.

[12] Karp, R.M.: Reducibility among Combinatorial Problems. In *Complexity of Computer Computations*, Miller, R.E., and Thatcher, J.W., eds., Plenum Press, New York, 1972.

[13] Knuth, D.: The Art of Computer Programming, Vol. 3: Searching and Sorting, Addison-Wesley, Reading, Mass., 1973.

[14] Kohler, W.H., and Steiglitz, K.: Characterization and Theoretical Comparison of Branch-and-Bound Algorithms for Permutation Problems. *Journal of the ACM*, Vol. 21 (1974), pp. 140-156.

[15] Lawler, E.L., and Wood, D.E.: Branch-and-Bound Methods: A survey. *Operations Research*, Vol. 14 (1966), pp. 699-719.

[16] Nauss, R.M.: The 0-1 Knapsack Problem with Multiple Choice Constraints, University of Missouri-St. Louis, 1975 (Revised in 1976).

[17] Teranaka, K.: An Approximate Algorithm for the Complementary Knapsack Programming Problem. Master Thesis, Department of Applied Mathematics and Physics, Kyoto University, 1976.

Toshihide IBARAKI: Department of
Applied Mathematics and Physics,
Kyoto University,
Kyoto, 606, Japan

Appendix. Modification of DUAL(P)

As mentioned at the end of Section 3 and after Theorem 4.2, it is desirable to find a numbering of β_k's with the same value such that Theorem 4.2 (b) can be applied. The exhaustive search of all possible numberings is, however, not computationally feasible since there are $K!$ different numberings if K β_k's take the same value. The following algorithm is heuristic, and may be considered as a compromize between these conflicting objectives.

Assume now that β_k's are initially numbered as in (3.23) and that

(A.1) $$\beta_{k'}, \beta_{k'+1}, \cdots, \beta_{k'+\ell}$$

take the same value. Let $J=\{k', k'+1, \ldots, k'+\ell\}$. We renumber these β_k's so that Theorem 4.2 (b) may be applied if possible. Partition J into two sets J_1 and J_2, such that

(A.2)
$$\beta_k \text{ is of } c_{im_i}/a_{im_i} \text{ type if } k\in J_1$$
$$\beta_k \text{ is of } (c_{ij-1}-c_{ij})/(a_{ij-1}-a_{ij}) \text{ type if } k\in J_2.$$

Assume further that

(A.3)
$$v'>0 \text{ holds for } k=k'+\ell+1,$$
$$v'\leq 0 \text{ holds for } k=k',$$

i.e., some β_k, $k\in J$, is selected as $\beta_{\bar{k}}$, when DUAL(P) is applied to these β_k's.

Denote $v'(>0)$ for $k=k'+\ell+1$ in DUAL(P) by v^*. The basic idea is as follows. First we check whether

(A.4) $$v^* - \Sigma\{a_{im_i} | c_{im_i}/a_{im_i} = \beta_k,\ k \in J_1\}$$

is positive or not. If it is nonpositive, choose $k \in J_1$ in any order and replace v' by $v'-a_{im_i}$ for each selected $\beta_k = c_{im_i}/a_{im_i}$, until $v' \leq 0$ is attained. Then Theorem 4.2 (b) is satisfied. On the other hand, if (A.4) is positive, we choose $k \in J_2$ in the FFD (First-Fit-Decreasing) order of $a_{ij-1}-a_{ij}$ (where $\beta_k = (c_{ij-1}-c_{ij})/(a_{ij-1}-a_{ij})$) whenever $v'-(a_{ij-1}-a_{ij}) \geq 0$ is satisfied. When β_k is chosen, v' is replaced by $v'-(a_{ij-1}-a_{ij})$. If $v' > 0$ holds at the end of this process, then $k \in J_1$ are chosen in any order (for each $k \in J_1$, v' is replaced by $v'-a_{im_i}$ if $\beta_k = c_{im_i}/a_{im_i}$), until $v' \leq 0$ is attained. If $v' \leq 0$ occurs in this process, Theorem 4.2 (b) is satisfied. On the other hand, if $v' > 0$ still holds after choosing all $k \in J_1$, a $k \in J_2$ rejected in the first stage (because of $v'-(a_{ij-1}-a_{ij}) < 0$) is chosen to yield $v' < 0$. If this is the case, Theorem 4.2 (b) is not satisfied.

This algorithm is based on the well known heuristic algorithm FFD for the bin-packing problem [11] (consider that a bin of length v^* is packed by items of length $a_{ij-1}-a_{ij}$).

Given index sets J, J_1, J_2 and $v^* > 0$ as above, the algorithm proceeds as follows.

M1. $v' \leftarrow v^*$, $k \leftarrow k'+1$, $J_3 \leftarrow \phi$. If $v^* - \Sigma\{a_{im_i} | c_{im_i}/a_{im_i} = \beta_k,\ k \in J_1\} \leq 0$, go to M3.

M2. Repeat the following procedure while $J_2 \neq \phi$: Select $k \in J_2$ with the maximum $a_{ij-1}-a_{ij}$ where $\beta_k = (c_{ij-1}-c_{ij})/(a_{ij-1}-a_{ij})$, and

$$J_2 \leftarrow J_2 - \{k\},$$

$$v' \leftarrow v' - (a_{ij-1}-a_{ij}) \text{ if } v'-(a_{ij-1}-a_{ij}) \geq 0$$

$$J_3 \leftarrow J_3 \cup \{k\} \text{ if } v'-(a_{ij-1}-a_{ij}) < 0.$$

M3. Repeat the following procedure while $v' > 0$ and $J_1 \neq \phi$: Select any $k \in J_1$ where $\beta_k = c_{im_i}/a_{im_i}$, and

$$J_1 \leftarrow J_1 - \{k\},\ v' \leftarrow v' - a_{im_i}.$$

M4. If $v' \leq 0$, halt; else select one $k \in J_3$ where $\beta_k = (c_{ij-1}-c_{ij})/(a_{ij-1}-a_{ij})$, and $v' \leftarrow v'-(a_{ij-1}-a_{ij})$ (<0). Halt.

It is now easy to see how this modification is incorporated in DUAL(P), and how an LP optimal solution of $\bar{P}$ is constructed. The detail is omitted.