韓國經營科學會誌
第26卷 第3號
2001年 9月


# 확장된 일반상한제약을 갖는 최대최소 선형계획 배낭문제*

원 중 연**

# The Maximin Linear Programming Knapsack Problem With Extended GUB Constraints

Joong-Yeon Won

■ Abstract ■

In this paper, we consider a maximin version of the linear programming knapsack problem with extended generalized upper bound (GUB) constraints. We solve the problem efficiently by exploiting its special structure without transforming it into a standard linear programming problem. We present an $O(n^3)$ algorithm for deriving the optimal solution where $n$ is the total number of problem variables. We illustrate a numerical example.

Keyword : Maximin Problem, LP Knapsack Problem, GUB Constraint, Cardinality Constraint, Computational Complexity

논문접수일 : 2001년 7월 11일 논문게재확정일 : 2001년 9월 5일
* This paper was supported by Kyonggi University Research Grants in 2000.
** Division of Advanced Industrial Engineering, Kyonggi University.

# 1. Introduction

In this paper, we consider the following maximin version of the linear programming (LP) knapsack problem with extended GUB constraints :

$$\text{(P) maximize } z=\min_{i\in M}\{\textstyle\sum_{j\in N_i} c_{ij}x_{ij}\} \quad (1)$$

$$\text{subject to } \textstyle\sum_{i\in M}\sum_{j\in N_i} a_{ij}x_{ij} \le b, \quad (2)$$

$$\textstyle\sum_{j\in N_i} x_{ij} \le k_i,\ i\in M, \quad (3)$$

$$0\le x_{ij}\le 1,\ j\in N_i,\ i\in M, \quad (4)$$

where $M=\{1,\cdots,m\}$ ; the classes $N_i=\{1,\cdots, n_i\}$ are mutually exclusive ; all $c_{ij}$, $a_{ij}$, $b$ and $k_i$ are positive integer numbers.

Problem (P) arises from the observation that for the public sector applications, the common utility can be frequently measured through maximin objective functions. Applications of (P) can be found in various areas : one example is arisen in the allocation of government funds to various departments. Each department has a number of projects requiring some allocations of funds. The knapsack constraint (2) restricts total budget to be spent. The extended GUB constraint (3) with (4) specifies the upper bound $k_i$ on the number of project to be funded in $i$th department. The objective (1) is to maximize the worst departmental sum of benefits accrued from the allocations of funds. This problem can be formulated as an integer version of (P). If this integer version is solved by a branch-and-bound procedure, we need to solve its LP relaxation of type (P).

The maximum sum version of (P) can be formulated as (P) by replacing the objective function (1) by

$$\text{maximize } z=\textstyle\sum_{i\in M}\sum_{j\in N_i} c_{ij}x_{ij}. \quad (5)$$

The maximum sum versions of (P) have been considered in Bagchi et al. [1] and Won [10] as extensions for the LP knapsack problem with ordinary GUB constraints. Several important applications of the maximum sum versions are given in Bagchi et al. [1]. Bagchi et al. [1] and Won [10] provide efficient algorithms of time complexity $O(n^2 \log n)$, where $n$ is the total number of problem variables.

The LP knapsack problem with ordinary GUB constraints can be formulated as (5), (2), (4) and replacing (3) by

$$\textstyle\sum_{j\in N_i} x_{ij}=1,\ i\in M. \quad (6)$$

Specialized algorithms for this problem have been developed in many papers, see the survey by Dudzinski and Walukiewicz [4]. Glover [6] provide $O(n\log n)$ algorithm for this problem. Dyer [5], Zemel [12], and Pisinger [9] provide $O(n)$ algorithms.

It is well known that problem (P) can be transformed into a standard LP problem by introducing $m$ additional constraints. However, it is not desirable computationally as it increases the size of problem (P) and furthermore the additional constraints affect the underlying special structure of (P).

In this paper, we suggest a solution algorithm that efficiently exploits the special structure of (P). In section 2, we first consider $m$ subproblems, which are obtained by allocating the resource $b$ to each subsystem of (P). The $i$th subsystem consists of all variables $x_{ij}$, $j\in N_i$, the corresponding part of the objective function, the knapsack constraint, and the extended GUB constraint. Subproblems can be extensions of the cardinality constrained LP knapsack problem [2, 3]. We identify some parametric properties in subproblems and present a parametric algorithm to derive the optimal objective function in changes of the allocated resource. The parametric algorithm has a time complexity of $O(n_i^2 \log n_i)$. In section 3, we first present an allocation problem (AP) which is equivalent to problem (P). The optimal solutions of (P) can easily be found from the optimal allocations of (AP). Then, we describe a solution algorithm for (P) by using the optimal objective functions derived from the subproblems in section 2. The solution algorithm for (P) has a time complexity of $O(n^3)$. Finally in section 4, we give a numerical example.

## 2. LP Knapsack Problem With Cardinality Constraint

In this section, we will consider the following subproblem (ECP*i*).

$$(\text{ECP}i)\quad z_i(b_i) = \text{maximize} \sum_{j\in N_i} c_{ij}x_{ij} \qquad (7)$$

$$\text{subject to} \sum_{j\in N_i} a_{ij}x_{ij} \le b_i, \qquad (8)$$

$$\sum_{j\in N_i} x_{ij} \le k_i, \qquad (9)$$

$$0 \le x_{ij} \le 1,\ j\in N_i, \qquad (10)$$

We will assume temporarily that the allocated resource $b_i$ is a known parameter. We also assume that the variables within each class $N_i$ are sorted so that $a_{ij_1} \le a_{ij_2}$ if $j_1 < j_2$, for each $i\in M$.

Subproblem (ECP*i*) is an extension of the cardinality constrained LP knapsack problem [2, 3]. The cardinality constrained LP knapsack problem (CP) can be formulated by replacing (9) by (6) in (ECP*i*). Campello and Maculan [2] discuss some properties of basic feasible solutions of (CP) and prove that there exists an $O(n_i^3)$ algorithm for solving (CP). Dudzinski [4] shows that (CP) can be solved in $O(n_i^2)$ by solving at most $n_i$ ordinary LP knapsack problems. However, their algorithms cannot be adapted to a parametric procedure for obtaining the optimal objective values as $b_i$ in (ECP*i*) is changed. We present a parametric algorithm to obtain the optimal objective function $z_i^*(b_i)$. This function will be used in section 3 to find the optimal solution of (P).

Note that a basic feasible solution to (CP) always has two fractional components [2]. However, observe that a basic feasible solution to (ECP*i*) can have one or two fractional components. (Refer to [11].) In both problems the sum of these two fractional components is always equal to 1.

Define the slope $\theta_i(j_1, j_2)$ associated with $x_{ij_1}$ and $x_{ij_2}$ as

$$\theta_i(j_1, j_2) \equiv (c_{ij_1} - c_{ij_2})/(a_{ij_1} - a_{ij_2}),$$
$$j_1 < j_2,\ j_1 \in N_i \cup \{0\},\ j_2 \in N_i. \quad (11)$$

For notational convenience, define $c_{i0} = a_{i0} \equiv 0$. In case of $j_1 = 0$ in equation (11), the slope $\theta_i(0, j_2) = c_{ij_2}/a_{ij_2}$ is associated with only one variable $x_{ij_2}$. The variable $x_{i0}$ can be a slack variable for constraint (9). If $a_{ij_1} = a_{ij_2}$, $\theta_i(j_1, j_2)$ is defined to be a large number.

Let $x^0$ be an optimal solution with all integer components, where $b_i$ is equal to $b_i^0 = \sum_{j\in N_i} a_{ij}x_{ij}^0$. Let $J_i$ be the index set of variables having the value 1 in a basic feasible solution to (ECP*i*). Observe that $|J_i| \le k_i$ always holds. The following theorem 1 identifies some parametric properties of (ECP$i$).

**Theorem 1.** As the value of $b_i$ increases from $b_i^0$ by a small amount, the optimal objective value of (ECP$i$) increases along the slope $\theta_i(f_1, f_2)$ determined by the following :

$$\theta_i(f_1, f_2) = \begin{cases} \max_{j \in N_i \setminus J_i} \{\theta_i(0, j)\}, & \text{if } |J_i| < k_i, \\ \max_{j_1 \in J_i} \max_{j_2 \in N_i \setminus J_i, j_2 > j_1} \{\theta_i(j_1, j_2)\}, & \text{if } |J_i| = k_i, \end{cases}$$

where $f_1$ and $f_2$ are indices of variables taking fractional values.

**Proof.** As the value of $b_i$ increases from $b_i^0$ by a small amount $\alpha$, the optimal objective value increases as follows :

$$z_i(b_i^0 + \alpha) = z_i(b_i^0) + \alpha c_B B^{-1} e$$

where B is a 2x2 basis matrix consisted of two column vectors from constraint (8) and (9), $c_B$ is a benefit vector corresponding to B, and $e = (1, 0)'$. There arise two cases. Case (i). If $|J_i| < k_i$, the current integer optimal solution $x^0$ satisfies the constraint (9) by strict inequality. So, one variable $x_{ij}$ belonging to $N_i \setminus J_i$ (currently having value 0) must be selected to have fractional value, while variables $x_{ij}$ belonging to $J_i$ (currently having value 1) should be fixed at their current values. The basis B for this case corresponds to a basic vector $(x_{ij}, x_{i0})$, where $x_{i0}$ is a slack variable for constraint (9), and $c_B = (c_{ij}, 0)$. Hence,

$$\begin{aligned} z_i(b_i^0 + \alpha) &= z_i(b_i^0) + \alpha(c_{ij}/a_{ij}) \\ &= z_i(b_i^0) + \alpha\theta_i(0, j). \end{aligned}$$

Therefore, the slope of maximal increase in objective value is determined by

$$\theta_i(f_1, f_2) = \max_{j \in N_i \setminus J_i} \{\theta_i(0, j)\}. \quad (12)$$

Case (ii). If $|J_i| = k_i$, the current integer optimal solution satisfies the constraint (9) by equality. So, one variable $x_{ij}$ belonging to $J_i$ (currently having value of 1) must decrease to have fractional value and one variable $x_{ij_2}$ such that $j_2 > j_1$ and $j_2 \in N_i \setminus J_i$ must increase from current value 0. Note that the sum of these two variables is equal to 1 and $a_{ij_1} < a_{ij_2}$. So, the basis B for this case corresponds to a basic vector $(x_{ij_1}, x_{ij_2})$ and $c_B = (c_{ij_1}, c_{ij_2})$. Hence,

$$\begin{aligned} z_i(b_i^0 + \alpha) &= z_i(b_i^0) + \alpha(c_{ij_1} - c_{ij_2})/(a_{ij_1} - a_{ij_2}) \\ &= z_i(b_i^0) + \alpha\theta_i(j_1, j_2). \end{aligned}$$

Therefore, the slope of maximal increase in objective value is determined by

$$\theta_i(f_1, f_2) = \max_{j_1 \in J_i} \max_{j_2 \in N_i \setminus J_i, j_2 > j_1} \{\theta_i(j_1, j_2)\} \quad (13)$$

From the equation (12) and (13), theorem 1 holds. ■

The maximal increment of $b_i$ where the new basis remains optimal is $a_{if_2} - a_{if_1}$. In case of $|J_i| < k_i$, the value of $x_{ij_2}$ reaches 1 as $b_i$ increases up to $a_{ij_2}$. (Note that $a_{if_1} = 0$ if $f_1 = 0$.) Then, the index $f_2$ is added to $J_i$. In case of $|J_i| = k_i$, the value of $x_{if_1}$ drops down to 0 and that of $x_{if_2}$ reaches 1, as $b_i$ increases up to $a_{if_2} - a_{if_1}$. So, the index $f_2$ is added to $J_i$ and $f_1$ is deleted from $J_i$.

It is well known that the optimal objective function $z_i^*(b_i)$ is a piecewise linear nondecreasing concave. So, $z_i^*(b_i)$ can be represented by a sequence of nonincreasing slopes $\theta_1(j_1, j_2)$. The magnitude of interval of $b_i$ corresponding to $\theta_1(j_1, j_2)$ is $a_{ij_2} - a_{ij_1}$. The following parametric algorithm selects only the optimal slopes from the list of all possible candidate slopes. The parametric algorithm terminates when there are no more slopes left in the candidate list, $CL_i$. Note also that we need only nonnegative slopes since the optimal objective values cannot decrease as $b_i$ increases. The output of parametric algorithm is the list $L_i$ of optimal slopes, which constitute the optimal objective function $z_i^*(b_i)$.

## Parametric Algorithm

Step 0. $CL_i \leftarrow \phi, L_i \leftarrow \phi, J_i \leftarrow \phi$.

Step 1. Reorder the variables according to nondecreasing $a_{ij}$ values so that $a_{ij_1} \leq a_{ij_2}$ whenever $j_1 < j_2$.
Compute all slopes $\theta_i(j_1, j_2) = (c_{ij_1} - c_{ij_2})/(a_{ij_1} - a_{ij_2})$, for $j_1 < j_2$, $j_1 \in N_i \cup \{0\}$, $j_2 \in N_i$,
where $c_{i0} = a_{i0} \equiv 0$. Put only nonnegative slopes into the candidate list $CL_i$ in nonincreasing order.

Step 2. Select the first slope $\theta_i(j_1, j_2)$ in $CL_i$.
If $CL_i = \phi$, stop the procedure. The list $L_i$ is the desired output.
If $|J_i| < k_i$, go to step 3. Otherwise, go to step 4.

Step 3. If $j_1 = 0$, set $L_i \leftarrow L_i \cup \{\theta_i(j_1, j_2)\}$ and $J_i \leftarrow J_i \cup \{j_2\}$. Go to step 5.

Step 4. If $j_1 \in J_i$ and $j_2 \in N_i \setminus J_i$, set $L_i \leftarrow L_i \cup \{\theta_i(j_1, j_2)\}$ and $J_i \leftarrow J_i \cup \{j_2\} \setminus \{j_1\}$.
Go to step 5.

Step 5. Set $CL_i \leftarrow CL_i \setminus \{\theta_i(j_1, j_2)\}$.
Go to step 2.

**Theorem 2.** The time complexity of parametric algorithm is $O(n_i^2 \log n_i)$.

**Proof.** Step 0 is an initialization and requires only constant effort. In step 1, reordering the variables can be accomplished in $O(n_i \log n_i)$ time. The computation of all slopes is carried out in $O(n_i^2)$ operations and the ordering of these slopes in $CL_i$ is accomplished in $O(n_i^2 \log n_i)$ time. So, step 1 requires overall $O(n_i^2 \log n_i)$ effort. Step 2 and step 5 need only constant effort. Each execution of step 2 must be followed by an execution of either step 3 or step 4, then by that of step 5. After each execution of step 5, one slope is deleted from $CL_i$. Since the total number of slopes in $CL_i$ (in step 2) is $O(n_i^2)$, step 3 and step 4 can be executed at most $O(n_i^2)$. Consider a particular execution of step 3 or step 4. In step 3, insertion of $j_2$ into $J_i$ in increasing order requires only $O(\log n_i)$ effort by using the binary search method [8]. In step 4, in identifying whether $j_1$ and $j_2$ belong to corresponding set or not, $O(\log n_i)$ examinations are required, and each insertion and deletion can be carried out in $O(\log n_i)$ operations by using the binary search method. So, the particular execution from step 2 to step 5 requires $O(\log n_i)$ effort and hence the main iterative steps, from step 2 to step 5, requires overall $O(n_i^2 \log n_i)$ efforts. Therefore, it follows that the overall complexity of parametric algorithm is $O(n_i^2 \log n_i)$. ■

## 3. The Solution Algorithm

Problem (P) can be decomposed into $m$ subproblems (ECP$i$), $i \in M$, by allocating the resource $b$ to each subsystem of (P). Therefore, problem (P) is equivalent to the following allocation problem (AP).

$$\text{(AP) maximize } z = \min_{i \in M} \{z_i(b_i)\}$$
$$\text{subject to } \sum_{i \in M} b_i \le b,$$
$$b_i \ge 0,\ i \in M,$$

where each objective function $z_i(b_i)$ can be obtained by applying the parametric algorithm (in section 2) to each subproblem (ECP$i$). To solve problem (P), we first find the optimal allocations $b_i^*$ of $b$ in (AP). Then, the optimal solution of (P) can easily be derived from the optimal allocations $b_i^*$, $i \in M$.

Observe that the optimal objective value of (AP) has the same value as that of each subproblem by the maximin objective in (AP) [8]. Therefore, the optimal allocations $b_i^*$ to each subproblem should be such that $z^* = z_1^* = \cdots = z_m^*$, where $z^*$ is the optimal objective values of (AP). In finding optimal allocations $b_i^*$, $i \in M$, we only need to keep track of the optimal slopes $\theta_i(j_1, j_2)$ from each $L_i$, $i \in M$. Now, we describe the solution algorithm for (P) in the following way.

### Solution Algorithm

Step 0. $\bar{z} \leftarrow 0$, $\bar{b} \leftarrow 0$, $J_i \leftarrow \phi$, $\Delta b_i \leftarrow 0$, $L_i \leftarrow \phi$, $i \in M$

Step 1. Obtain lists $L_i$, $i \in M$, by applying the parametric algorithm to each subproblem (ECP$i$).

Step 2. Choose the first slope $\theta_i(j_1, j_2)$ from each $L_i$, $i \in M$ . Set $\Delta b_i = a_{ij_2} - a_{ij_1}$, $i \in M$.

Step 3. Compute $\Delta z_i = \theta_i(j_1, j_2) * \Delta b_i$, $i \in M$,
$\Delta z_{\bar{i}} = \min_{i \in M} \{\Delta z_i\}$,
$\bar{i} = \arg\min_{i \in M} \{\Delta z_i\}$, $\bar{z} = \bar{z} + \Delta z_{\bar{i}}$.
If there are several $\Delta z_{\bar{i}}$, choose one arbitrarily.

Step 4. Compute $\bar{b} = \bar{b} + \sum_{i \in M} \{\Delta z_{\bar{i}} / \theta_i(j_1, j_2)\}$.

Step 5. If $\bar{b} \ge b$, compute the optimal objective value $z^*$ as
$z^* = \bar{z} - (\bar{b} - b) / \{\sum_{i \in M} \{1/\theta_i(j_1, j_2)\}\}$.
Go to step 8, and find an optimal solution to (P).
Otherwise, compute
$\Delta b_i = \Delta b_i - \Delta z_{\bar{i}} / \theta_i(j_1, j_2)$, $i \in M$.
Set $L_{\bar{i}} \leftarrow L_{\bar{i}} \backslash \{\theta_{\bar{i}}(j_1, j_2)\}$, and go to step 6.

Step 6. If $L_{\bar{i}} = \phi$, the optimal objective value $z^*$ is $\bar{z}$. Go to step 8 and find an optimal solution to (P).
Otherwise, update $J_{\bar{i}}$ as following and go to step 7.
If $|J_{\bar{i}}| < k_{\bar{i}}$, $J_{\bar{i}} \leftarrow J_{\bar{i}} \cup \{j_2\}$,
Otherwise, $J_{\bar{i}} \leftarrow J_{\bar{i}} \cup \{j_2\} \backslash \{j_1\}$.

Step 7. Choose the first slope $\theta_{\bar{i}}(j_1, j_2)$ from $L_{\bar{i}}$.
Compute $\Delta b_{\bar{i}} \leftarrow a_{\bar{i}j_2} - a_{\bar{i}j_1}$. Go to step 3.

Step 8. The optimal solution to (P) can be obtained as follows :

For $i$ such that $j_1 \neq 0$ in $\theta_i(j_1, j_2)$,

$$x_{ij_1} = (z^* - \sum_{j \in J_i \setminus \{j_1\}} c_{ij} - c_{ij_2}) / (c_{ij_1} - c_{ij_2}),$$

$$x_{ij_2} = 1 - x_{ij_1},$$

$$x_{ij} = 1,\ j \in J_i \setminus \{j_1\},$$

$$x_{ij} = 0,\ j \in N_i \setminus J_i \setminus \{j_2\}.$$

For $i$ such that $j_1 = 0$ in $\theta_i(j_1, j_2)$,

$$x_{ij_2} = (z^* - \sum_{j \in J_i} c_{ij}) / c_{ij_2},$$

$$x_{ij} = 1,\ j \in J_i,$$

$$x_{ij} = 0,\ j \in N_i \setminus J_i \setminus \{j_2\}.$$

The solution algorithm starts the allocation procedure from $b_i = 0$, $i \in M$, and hence $z = 0$. Next, it chooses the first slope $\theta_i(j_1, j_2)$ that is the largest one in each $L_i$, $i \in M$. Then, it increases the objective value $z$ along the chosen slopes $\theta_i(j_1, j_2)$ until a break point with the smallest objective value $\Delta z_{\bar{i}} = \min_{i \in M}\{\Delta z_i\}$ is reached, where $\Delta z_i = \theta_i(j_1, j_2) * (a_{ij_2} - a_{ij_1})$, $i \in M$. Note that each $z_i(b_i)$ is a piecewise linear nondecreasing concave function with a number of break points. The additional allocations to each subproblem are computed by $\Delta z_{\bar{i}} / \theta_i(j_1, j_2)$, $i \in M$. If the current total allocation $\bar{b} = \sum_{i \in M} \Delta z_{\bar{i}} / \theta_i(j_1, j_2)$ is greater than or equal to $b$, the optimal solutions to (P) can be derived in this step. Otherwise, it increases the objective value $z$ again until the next break point is reached. And then, it iterates the same procedures.

**Theorem 3.** The time complexity of the main algorithm for (P) is $O(n^3)$.

**Proof.** Step 0 requires constant effort. In step 1, each list $L_i$ can be obtained by applying the parametric algorithm in $O(n_i^2 \log n_i)$ time, totaling $\sum_{i \in M} O(n_i^2 \log n_i) \leq O(n^2 \log n)$ time. In step 2, choosing each largest slopes from all $L_i$, $i \in M$, and computing $\Delta b_i$, $i \in M$, require overall $O(m)$ effort. In step 3, $\Delta z_i$, $i \in M$, $\Delta z_{\bar{i}}$ and $\bar{z}$ can be computed in $O(m)$ operations and comparisons. In step 4, $\bar{b}$ is computed in $O(m)$ operations. In step 5, computations of $z^*$, $\Delta b_i$, and deletion of the slope from list $L_{\bar{i}}$ can be carried out in $O(m)$ operations. In step 6, insertion and deletion can be carried out in $O(\log n_{\bar{i}}) \leq O(\log n)$ operations by using the binary search method [8]. Step 7 requires constant effort. So, particular executions from step 3 to step 7 require $\max\{O(m), O(\log n)\} \leq O(n)$ effort. Since the total number of slopes in all lists is $O(n^2)$ and each execution of step 5 delete one slope from some list, executions from step 3 to step 7 can be run at most $O(n^2)$ times. So, the total iterations from step 3 to step 7 require $O(n^3)$ efforts. The optimal solution in step 8 can be obtained in $O(n)$ operations. Therefore, it follows that the overall complexity of main algorithm is $\max\{O(n^2 \log n), O(n^3)\} \leq O(n^3)$ effort. ∎

## 4. Numerical Example

Consider the maximin LP knapsack problem with two extended GUB constraints, where $M = \{1, 2\}$, $N_i = \{1, \ldots, 6\}$, $i \in M$, $b = 20$, $k_1 = 2$, $k_2 = 2$. The values of $c_{ij}$ and $a_{ij}$ are as follows :

| i | 1 | | | | | | 2 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| j | 1 | 2 | 3 | 4 | 5 | 6 | 1 | 2 | 3 | 4 | 5 | 6 |
| $c_{ij}$ | 7 | 5 | 12 | 11 | 15 | 17 | 3 | 8 | 11 | 9 | 13 | 14 |
| $a_{ij}$ | 2 | 4 | 5 | 8 | 9 | 13 | 1 | 2 | 5 | 6 | 9 | 12 |

<Iteration 1>

Step 0. $\bar{z} \leftarrow 0,\ \bar{b} \leftarrow 0,\ J_i \leftarrow \phi,\ \Delta b_i \leftarrow 0,$
$L_i \leftarrow \phi,\ i \in \{1,2\}.$

Step 1. $L_1 = \{\ \theta_1(0,1) = 3.5,\ \ \theta_1(0,3) = 2.4,$
$\theta_1(1,5) = 1.143,\ \theta_1(3,6) = 0.625\}$
$L_2 = \{\ \theta_2(0,2) = 4,\ \ \theta_2(0,1) = 3,$
$\theta_2(1,3) = 2,\ \ \theta_2(2,5) = 0.714,$
$\theta_2(3,6) = 0.429\}.$

Step 2. $\theta_1(0,1) = 3.5$ is selected from $L_1$,
$\theta_2(0,2) = 4$ is selected from $L_2$,
$\Delta b_1 = a_{11} - a_{10} = 2,\ \ \Delta b_2 = a_{22} - a_{20} = 2.$

Step 3. $\Delta z_1 = \theta_1(0,1) * \Delta b_1 = 7,$
$\Delta z_2 = \theta_2(0,2) * \Delta b_2 = 8,$
$\Delta z_{\bar{i}} = \min\{7,8\} = 7(= \Delta z_i),\ \ \bar{i} = 1,$
$\bar{z} = \bar{z} + \Delta z_{\bar{i}} = 7.$

Step 4. $\bar{b} = 0 + 7/3.5 + 7/4 = 3.75.$

Step 5. Since $\bar{b} < b,\ \ \Delta b_1 = 2 - 7/3.5 = 0,$
$\Delta b_2 = 2 - 7/4 = 0.25,\ L_1 \leftarrow L_1 \backslash \{\theta_1(0,1)\}.$

Step 6. Since $L_1 \neq \phi$ and $|J_1|\ (=0) < k_1(=2),$
$J_1 \leftarrow \{1\}.$

Step 7. $\theta_1(0,3) = 2.4$ is selected from $L_1$,
$\Delta b_1 = a_{13} - a_{10} = 5$, go to step 3.

<Iteration 2>

Step 3. $\Delta z_1 = \theta_1(0,3) * \Delta b_1 = 12,$
$\Delta z_2 = \theta_2(0,2) * \Delta b_2 = 1,$
$\Delta z_{\bar{i}} = \min\{12,1\} = 1(= \Delta z_2),\ \ \bar{i} = 2,$
$\bar{z} = 7 + 1 = 8.$

Step 4. $\bar{b} = 3.75 + 1/2.4 + 1/4 = 4.417.$

Step 5. Since $\bar{b} < b,\ \ \Delta b_1 = 5 - 1/2.4 = 4.583,$
$\Delta b_2 = 0.25 - 1/4 = 0,\ L_2 \leftarrow L_2 \backslash \{\theta_2(0,2)\}.$

Step 6. Since $L_2 \neq \phi$ and $|J_2|(=0) < k_2(=2),$
$J_2 \leftarrow \{2\}.$

Step 7. $\theta_2(0,1) = 3$ is selected from $L_2$,
$\Delta b_2 = a_{21} - a_{20} = 1$, go to step 3.

<Iteration 3>

Step 3. $\Delta z_1 = \theta_1(0,3) * \Delta b_1 = 11,$
$\Delta z_2 = \theta_2(0,1) * \Delta b_2 = 3,$
$\Delta z_{\bar{i}} = \min\{11,3\} = 3(= \Delta z_2),\ \ \bar{i} = 2,$
$\bar{z} = 8 + 3 = 11.$

Step 4. $\bar{b} = 4.417 + 3/2.4 + 3/3 = 6.667.$

Step 5. Since $\bar{b} < b,\ \ \Delta b_1 = 4.583 - 3/2.4 = 3.33,$
$\Delta b_2 = 1 - 3/3 = 0,\ \ L_2 \leftarrow L_2 \backslash \{\theta_2(0,1)\}.$

Step 6. Since $L_2 \neq \phi$ and $|J_2|(=1) < k_2(=2),$
$J_2 \leftarrow \{1,2\}.$

Step 7. $\theta_2(1,3) = 2$ is selected from $L_2$,
$\Delta b_2 = a_{23} - a_{21} = 5 - 1 = 4$, go to step 3.

<Iteration 4>

Step 3. $\Delta z_1 = \theta_1(0,3) * \Delta b_1 = 8,$
$\Delta z_2 = \theta_2(1,3) * \Delta b_2 = 8,$
$\Delta z_{\bar{i}} = \min\{8,8\} = 8(= \Delta z_1$ or $\Delta z_2),$
$\bar{i} = 1,\ \ \bar{z} = 11 + 8 = 19.$

Step 4. $\bar{b} = 6.667 + 8/2.4 + 8/2 = 14.$

Step 5. Since $\bar{b} < b,\ \ \Delta b_1 = 3.33 - 8/2.4 = 0,$
$\Delta b_2 = 4 - 8/2 = 0,\ \ L_1 \leftarrow L_1 \backslash \{\theta_1(0,3)\}.$

Step 6. Since $L_1 \neq \phi$ and $|J_1|(=1) < k_1(=2),$
$J_2 \leftarrow \{1,3\}.$

Step 7. $\theta_1(1,5) = 1.143$ is selected from $L_1$,

$\Delta b_1 = a_{15} - a_{11} = 9 - 2 = 7$, go to step 3.

<Iteration 5>

Step 3. $\Delta z_1 = \theta_1(1,5) * \Delta b_1 = 8$,

$\Delta z_2 = \theta_2(1,3) * \Delta b_2 = 0$,

$\Delta z_{\bar{i}} = \min\{8, 0\} = 0 (= \Delta z_2)$, $\bar{i} = 2$,

$\bar{z} = 19 + 0 = 19$.

Step 4. $\bar{b} = 14 + 0/1.143 + 0/2 = 14$.

Step 5. Since $\bar{b} < b$, $\Delta b_1 = 7 - 0/1.143 = 7$,

$\Delta b_2 = 0 - 0/2 = 0$, $L_2 \leftarrow L_2 \backslash \{\theta_2(1,3)\}$.

Step 6. Since $L_2 \neq \phi$ and $|J_2| (= 2) = k_2 (= 2)$,

$J_2 \leftarrow \{2, 3\}$.

Step 7. $\theta_2(2,5) = 0.714$ is selected from $L_2$,

$\Delta b_2 = a_{25} - a_{22} = 9 - 2 = 7$, go to step 3.

<Iteration 6>

Step 3. $\Delta z_1 = \theta_1(1,5) * \Delta b_1 = 8$,

$\Delta z_2 = \theta_2(2,5) * \Delta b_2 = 5$,

$\Delta z_{\bar{i}} = \min\{8, 5\} = 5 (= \Delta z_2)$, $\bar{i} = 2$,

$\bar{z} = 19 + 5 = 24$.

Step 4. $\bar{b} = 14 + 5/1.143 + 5/0.714 = 25.38$.

Step 5. Since $\bar{b} (= 25.38) > b (= 20)$, the optimal objective value is

$$z^* = 24 - (25.38 - 20)/(1/1.143 + 1/0.714) = 21.636.$$

Step 8. The optimal solution to (P) is as follows :

$x_{11} = (21.636 - 12 - 15)/(7 - 15) = 0.67$,

$x_{15} = 1 - 0.67 = 0.33$,

$x_{13} = 1$, $x_{1j} = 0$, $j = 2, 4, 6$,

$x_{22} = (21.636 - 11 - 13)/(8 - 13) = 0.473$,

$x_{25} = 1 - 0.473 = 0.527$,

$x_{23} = 1$, $x_{2j} = 0$, $j = 1, 4, 6$,

# 5. Conclusions

This paper suggests a maximin version (P) for the linear programming knapsack problem with extended generalized upper bound constraints [1, 10]. To solve problem (P), we introduce a parametric allocation of the knapsack right-hand side in (P). Each subproblem obtained from the parametric allocation is an extension of the cardinality constrained linear programming knapsack problem [2, 3].

First, we identify some parametric properties in each subproblem and then describe a parametric algorithm to obtain the optimal objective function of each subproblem as the knapsack allocation is changed. The parametric algorithm has the time complexity of $O(n_i^2 \log n_i)$, where $n_i$ is the number of variables in each subproblem. Next, using the optimal objective functions of subproblems, we have developed the solution algorithm for (P). The time complexity of solution algorithm is of order $O(n^3)$, where $n$ is the total number of problem variables.

# References


[1] Bagchi, A., Bhattacharyya, N., and Chakravarti, N., "LP relaxation of the Two Dimensional Knapsack Problem with Box and GUB constraints," *European Journal of Operational Research*, 89(1996), pp.609-617.

[2] Campello, R.E. and Maculan, N.F., "Lagrangean Relaxation for a Lower Bound to a Set Partitioning Problem with Side Constraints : Properties and Algorithms," *Discrete Applied Mathematics*, 18(1987), pp.119-136.

[3] Dudzinski, K., "On a Cardinality Constra-

ined Linear Programming Knapsack Problem," *Operations Research Letters*, 8(1989), pp.215-218.

[4] Dudzinski, K. and Walukiewicz, S., "Exact Methods for the Knapsack Problem and its Generalizations," *European Journal of Operational Research*, 28(1987), pp.3-21.

[5] Dyer, M.E., "An O(n) Algorithm for the Multiple-Choice Knapsack Linear Program," *Mathematical Programming*, 29(1984), pp.57-63.

[6] Glover, F. and Klingman, D., "An O(nlogn) algorithm for the LP Knapsacks with GUB Constraints," *Mathematical Programming*, 29 (1979), pp.345-361.

[7] Kaplan, S., "Applications of Programs with Maximin Objective Functions to Problems of Optimal Resource Allocation," *Operations Research*, 22(1974), pp.802- 807.

[8] Kronsjö, L.I., *Algorithms : Their Complexities and Efficiency*, Wiley, Chichester, 1979.

[9] Pisinger, D., "A Minimal Algorithm for the Multiple-Choice Knapsack Problem," *European Journal of Operational Research*, 83 (1995), pp.394-410.

[10] Won, J.Y., "An $O(n^2 \log n)$ Algorithm for the Linear Knapsack Problem with SUB and Extended GUB Constraints," *Journal of the Korean OR/MS Society*, 22(1997), pp.1-9.

[11] Won, J.Y., "On a Two Dimensional Linear Programming Knapsack Problem with the Extended GUB constraint," *Journal of the Korean Institute of Industrial Engineers*, 27 (2001), pp.25-29.

[12] Zemel, E., "An O(n) Algorithm for the Linear Multiple-Choice Knapsack Problem and Related Problems," *Information Processing Letters*, 18(1984), pp.132-128.