AN ALGORITHM FOR SOLVING
BILINEAR KNAPSACK PROBLEMS

Hiroshi Konno
University of Tsukuba

(Received May 9, 1981)

Abstract This paper proposes a cutting plane algorithm for solving 0-1 bilinear knapsack problem (BK), a special kind of integer quadratic programming problem in which a bilinear function $c^T x + d^T y + x^T C y$ is to be maximized subject to disjoint knapsack constraints on $x$ and $y$. This problem has applications in bipartite matching, multi-attribute utility analysis, cutting stock problems, and so on.

The main purpose of this paper is to demonstrate the practicality of cutting plane algorithm which is an elaboration of similar algorithms proposed for bilinear linear programming problems and convex maximization problems. It will be shown that a very effective cut can be generated without excessive amount of computation and that our approach is potentially advantageous over the standard linearization approach.

1. Introduction

This paper proposes a finitely convergent cutting plane algorithm for solving 0-1 bilinear knapsack problem (BK), a special type of 0-1 integer quadratic programming problem to be defined below:

$$\begin{align*}
\text{maximize} \quad & \phi(x, y) = \sum_{i=1}^{m} c_i x_i + \sum_{j=1}^{n} d_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} x_i y_j \\
\text{subject to} \quad & \sum_{i=1}^{m} a_{i} x_i \leq a_0, \quad x_i \in \{0, 1\}, \quad i = 1, \ldots, m, \\
& \sum_{j=1}^{n} b_{j} y_j \leq b_0, \quad y_j \in \{0, 1\}, \quad j = 1, \ldots, n.
\end{align*}$$

(BK)

This problem has applications in bipartite matching problems, cutting stock problems, multi-attribute utility analysis to name only a few. BK is the simplest discrete analogue of the bilinear linear programming problem (BL):
Algorithm for Bilinear Knapsack Problems

\[
\begin{align*}
\text{maximize} \quad & \sum_{i=1}^{m} c_i x_i + \sum_{j=1}^{n} d_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_i y_j \\
\text{subject to} \quad & \sum_{i=1}^{m} a_{ir} x_i \leq a_r, \quad r=1, \ldots, k; \quad x_i \geq 0, \quad i=1, \ldots, m, \\
& \sum_{j=1}^{n} b_{sj} y_j \leq b_s, \quad s=1, \ldots, l; \quad y_j \geq 0, \quad j=1, \ldots, n.
\end{align*}
\]

which has attracted more attention in recent years. For example, the present author proposed a cutting plane algorithm [8] and showed that it pertains to many real world applications [9]. Readers are referred to [3, 4, 8, 13, 15] for algorithms and to [2, 9, 12, 14] for applications of BL.

It is true that BK can be reformulated as a standard 0-1 integer linear program by introducing new variables and constraints [6]. But this manipulation usually increases the size of the problem and destroys the problem structure.

The algorithm to be developed in this paper, on the other hand, directly exploits the special structure of the problem like the ones proposed by the present author for BL [8] and for convex maximization problem [10]. This algorithm consists of two subprocedures. One is to obtain a local maximum which amounts to solving a sequence of 0-1 knapsack problems and small scale 0-1 integer linear programs. The other is to adjoin a cutting plane which eliminates a local maximum and yet does not eliminate any solution potentially better than the current incumbent. It will be shown in Section 3 that cut coefficients can be obtained by solving parametric knapsack problems (or parametric linear programming problems). Also, finite convergence of the algorithm is established without using expensive cuts such as disjunctive face cut [12].

Though this algorithm can be easily extended to general 0-1 bilinear integer programs (e.g., bilinear assignment problem), we will concentrate here on BK because (i) it is one of the most important problems in this class and (ii) its structure pertains to various algorithmic elaboration. Numerical results on this algorithm for the problems of the size up to \( m=20, n=100 \) will be reported in Section 4. All the test problems were solved successfully within a reasonable amount of time.

2. Alternate Mountain Climbing to Obtain a Local Star Maximum

Consider a 0-1 bilinear knapsack problem:
362

(2.1) \[
\begin{aligned}
\text{maximize } & \phi(x, y) = \sum_{i=1}^{m} a_i x_i + \sum_{j=1}^{n} d_j y_j + \sum_{i=1,j=1}^{m,n} c_{ij} x_i y_j \\
\text{subject to } & x \in X_0, \ y \in Y_0.
\end{aligned}
\]

where

\begin{align}
(2.2) & \quad X_0 = \{x \in \mathbb{R}^m \mid \sum_{i=1}^{m} a_i x_i \leq \alpha_0, \ x_i \in \{0, 1\}, \ i=1, \ldots, m\} \\
(2.3) & \quad Y_0 = \{y \in \mathbb{R}^n \mid \sum_{j=1}^{n} b_j y_j \leq \beta_0, \ y_j \in \{0, 1\}, \ j=1, \ldots, n\}
\end{align}

It will be assumed throughout that \(a_i's\) and \(b_j's\) are positive integers and that \(c_{ij}'s, d_j's\) and \(c_{ij}'s\) are (not necessarily positive) integers. Also, we will assume that \(X_0 \neq \emptyset, Y_0 \neq \emptyset\) and that \((x^*, y^*)\) is the current incumbent, i.e., the best feasible solution of (2.1) identified to date.

A natural step to obtain a local maximum is to maximize \(\phi\) by alternately fixing the value of \(x\) or \(y\), which amounts to solving a sequence of knapsack problems. To simplify the notation, let

\begin{align}
(2.4) & \quad K_X(y): \max \{\phi(x, y) \mid x \in X\} \\
(2.5) & \quad K_Y(x): \max \{\phi(x, y) \mid y \in Y\}
\end{align}

The set \(X \subset X_0\) is initially taken to be \(X_0\), in which cases \(K_X(y)\) as well as \(K_Y(x)\) is a 0-1 knapsack problem. Note, however, that constraints will be added to \(X_0\) in the later steps of the algorithm, so that \(K_X(y)\) will, in general, be a 0-1 integer program without special structures.

Procedure AMC(X, Y_0; x^0, y^0) (Alternate Mountain Climbing)

Step 0. Given \(x^0 \in X_0\) and \(y^0 \in Y_0\), let \(k=1\).

Step 1. Obtain an optimal solution \(x^k\) of \(K_X(y^{k-1})\) and an optimal solution \(y^k\) of \(K_Y(x^k)\).

Step 2. If \(\phi(x^k, y^k) > \phi(x^{k-1}, y^{k-1})\), then let \(k=k+1\) and go to Step 1. Otherwise, let \((\bar{x}, \bar{y}) = (x^k, y^k)\) and go to Step 3.

Step 3. If \(\phi(\bar{x}, \bar{y}) > \phi(x^*, y^*)\), then let \((x^*, y^*) = (\bar{x}, \bar{y})\) and halt.

The pair of points, \((\bar{x}, \bar{y})\) defined above will be called a locally maximal pair.
Theorem 1. Procedure AMC($X, Y_0; y^0$) generates a locally maximal pair in finitely many steps.

Proof: Follows from the finiteness of $X$ and $Y_0$ and from the monotonically increasing property of the sequence $\phi(x^k, y^k)$. ||

Once a locally maximal pair is reached, one cannot improve the objective function by fixing either $x$ equal to $\tilde{x}$ or $y$ equal to $\tilde{y}$. In this case, one will switch to a semi-global optimization process.

Let $x(i)$ be the $i$-th complement of $x \in X_0$, i.e.,

$$x(i) = (x_1, \ldots, x_{i-1}, 1-x_i, x_{i+1}, \ldots, x_m)$$

and let

$$I(x) = \{i \mid x(i) \in X\}.$$

Definition. $(\hat{x}, \hat{y})$ is a local star maximum if $(\hat{x}, \hat{y})$ is a locally maximal pair and

$$\max_i \{\phi(x(i), y) \mid y \in Y_0\} \leq \phi(\hat{x}, \hat{y})$$

holds for all $i \in I(\hat{x})$. ||

It is easy to see that the procedure defined below generates a local star maximum in finitely many steps.

Procedure SGO ($X, Y_0$) (Semi-Global Optimization)

Step 0. Choose $y^0 \in Y_0$ arbitrarily.

Step 1. Execute AMC($X, Y_0; x^0, y^0$) and let $(\hat{x}, \hat{y})$ be a locally maximal pair.

Step 2. If $\max_i \{\phi(x(i), y) \mid y \in Y_0\} \leq \phi(\hat{x}, \hat{y})$, $\forall i \in I(\hat{x})$, then halt.

Otherwise, go to Step 3.

Step 3. Let $y^0 = \tilde{y}$ where $\tilde{y} \in Y_0$ satisfies $\phi(x(i), \tilde{y}) > \phi(\hat{x}, \hat{y})$ for some $i \in I(\hat{x})$. Go to Step 1.

3. Cutting Planes from a Local Star Maximum

Given a local star maximum $(\hat{x}, \hat{y})$, the next step is to introduce a cutting plane which eliminates $\hat{x}$ and yet does not eliminate any $x \in X_0$ for which

$$\max \{\phi(x, y) \mid y \in Y_0\} > \phi(x^*, y^*)$$
Such a cutting plane will be called "valid". Note that (3.1) is tantamount to
\[
\max \{ \phi(x, y) \mid y \in Y_0 \} \geq \phi(x^*, y^*) + 1
\]
since \( \sigma_i's, d_j's, \sigma_{ij} 's \) as well as \( x_i 's \) and \( y_j 's \) are integers.

For simplicity, let
\[
z_i = \begin{cases} 
  x_i & \text{if } \bar{z}_i = 0 \\
  1 - x_i & \text{if } \bar{z}_i = 1
\end{cases} 
\]
so that \( z = 0 \) corresponds to \( \bar{x} \).

Let \( \psi(z, y) \) be the expression of \( \phi(x, y) \) relative to a new set of variables, i.e.,
\[
\phi(x, y) = \sum_{i=1}^{m} \sigma_i x_i + \sum_{j=1}^{n} d_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} \sigma_{ij} x_i y_j
\]
\[
= \sum_{i=1}^{m} \gamma_i x_i + \sum_{j=1}^{n} \delta_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} \gamma_{ij} x_i y_j + \phi(\bar{x}, 0)
\]
\[
= \psi(z, y)
\]
Also, let
\[
g_i(\lambda) = \max \{ \psi(z, y) \mid 0 \leq z, y \leq e^i, \ y \in Y_0 \}, \ i = 1, \ldots, m
\]
where \( \lambda \geq 0 \) and \( e^i \) represents the \( i \)-th unit vector. Note that
\[
g_i(0) = \phi(\bar{x}, \bar{y})
\]
since by definition,
\[
g_i(0) = \max \{ \psi(0, y) \mid y \in Y_0 \} = \max \{ \phi(\bar{x}, y) \mid y \in Y_0 \} = \phi(\bar{x}, \bar{y})
\]

Lemma 2: \( g_i \) is convex on \([0, \infty)\) for all \( i \).

Proof: The maximum of the right hand side of (3.5) is attained either at \( z = 0 \) or at \( z = \lambda e^i \) since \( \psi \) is linear in \( z \). Hence we have by (3.6)
\[
g_i(\lambda) = \max \{ \phi(\bar{x}, \bar{y}), \ h_i(\lambda) \}
\]
where
\[
h_i(\lambda) = \max \{ \sum_{j=1}^{n} (\delta_j + \gamma_{ij}) y_j \mid y \in Y_0 \} + \gamma_i \lambda + \phi(\bar{x}, 0)
\]
It is straightforward to see that \( h_i \) is convex, hence \( g_i \) is convex via (3.7).

Given \( g_i \), let
\[
\bar{\lambda}_i = \max \{ \lambda \mid g_i(\mu) \leq \phi(x^*, y^*) + 1, \ \forall \mu \in [0, \lambda] \}
\]
Algorithm for Bilinear Knapsack Problems

Lemma 3: $\hat{\lambda}_i > 0$ for all $i$. Also $\hat{\lambda}_i \leq 1$ for all $i$ such that $\hat{x}^{(i)} \in X_0$.

Proof: $\hat{\lambda}_i > 0$ follows from the definition (3.9) by noting the relation (3.6) and convexity of $g_i$. If $\hat{x}^{(i)} \in X$, then

$$h_i(1) = \max \{ \phi(\hat{x}^{(i)}, y) \mid y \in Y_0 \} \leq \phi(\hat{x}, \hat{y})$$

since $(\hat{x}, \hat{y})$ is a local star maximum. Also if $\hat{x}^{(i)} \in X_0 - X$, then

$$h_i(1) = \max \{ \phi(\hat{x}^{(i)}, y) \mid y \in Y_0 \} \leq \phi(x^*, y^*) + 1$$

since $\hat{x}^{(i)}$ has been cut off by a valid cut adjoined earlier. Therefore,

$$g_i(1) = \max \{ \phi(\hat{x}, \hat{y}), h_i(1) \} \leq \phi(x^*, y^*) + 1$$

for all $i$ such that $\hat{x}^{(i)} \in X_0$. The lemma follows from this by noting (3.6) and the convexity of $g_i$.

Let us define mutually exclusive sets of indices:

(3.10) $I = \{ i \mid \hat{\lambda}_i < \infty \}$

(3.11) $J = \{ i \mid \hat{\lambda}_i = \infty \}$

For $i \in J$, let

(3.12) $G_i(\mu) = \min \{ \psi(z, y) \mid -\mu e_i \leq z \leq 0; y \in Y_0 \}$

(3.13) $\hat{\mu}_i = \max \{ \mu \mid G_i(\mu) \leq \phi(x^*, y^*) + 1 \}$

Lemma 4: $\hat{\mu}_i > 0$ for all $i \in J$.

Proof: Follows from the continuity of $G_i$ and from $G_i(0) = \phi(\hat{x}, \hat{y}) \leq \phi(x^*, y^*) + 1$.

Theorem 5:

(3.14) $\sum_{i \in I} \frac{z_i}{\hat{\lambda}_i} - \sum_{i \in J} \frac{z_i}{\hat{\mu}_i} \geq 1$

is a valid cut.

Proof: This can be proved by applying a general theorem established in [7], but we will give an elementary proof for completeness. Let

(3.15) $Z = \{ z \in \mathbb{R}^m \mid \sum_{i \in I} \frac{z_i}{\hat{\lambda}_i} - \sum_{i \in J} \frac{z_i}{\hat{\mu}_i} \leq 1, z_i \geq 0, i = 1, \ldots, m \}$

One needs to show

(3.16) $\max \{ \psi(z, y) \mid y \in Y_0 \} \leq \phi(x^*, y^*) + 1, \forall z \in Z$

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.
It is easy to see that the extreme points of \( Z \) are given by
\[
z^i = \lambda_i e^i, \quad i \in I
\]
and the extreme directions of \( Z \) are given by
\[
(1) \quad e^j, \quad j \in J.
\]
\[
(2) \quad \mu_j e^j + \lambda_i e^i, \quad j \in J, \quad i \in I.
\]
where \( e^i \) and \( e^j \) are the \( i \)-th and \( j \)-th unit vector, respectively. Hence \( z \in Z \) can be expressed as
\[
z = \sum_{i \in I} \theta_i z^i + \sum_{j \in J} \alpha_j e^j + \sum_{i \in I} \sum_{j \in J} \alpha_{i,j} (\mu_j e^j + \lambda_i e^i)
\]
where
\[
\sum_{i \in I} \theta_i = 1, \quad \theta_i \geq 0, \quad \forall i \in I; \quad \alpha_j \geq 0, \quad \forall j \in J; \quad \alpha_{i,j} \geq 0, \quad \forall i \in I, \quad \forall j \in J
\]
so that
\[
\psi(x, y) = \sum_{i \in I} \theta_i \psi(z^i, y) + \sum_{j \in J} \alpha_j \psi(e^j, y) + \sum_{i \in I} \sum_{j \in J} \alpha_{i,j} \psi(\mu_j e^j + \lambda_i e^j, y)
\]
Hence
\[
\max \{ \psi(x, y) \mid y \in Y_0 \} \leq \sum_{i \in I} \theta_i \max \{ \psi(z^i, y) \mid y \in Y_0 \} + \sum_{j \in J} \alpha_j \max \{ \psi(e^j, y) \mid y \in Y_0 \}
\]
\[
+ \sum_{i \in I} \sum_{j \in J} \alpha_{i,j} \max \{ \psi(\mu_j e^j + \lambda_i e^j, y) \mid y \in Y_0 \}
\]
The first term of the right hand side is less than \( \phi(x^*, y^*) + 1 \) since
\[
\max \{ \psi(z^i, y) \mid y \in Y_0 \} \leq \phi(x^*, y^*) + 1, \quad \forall i \in I, \quad \text{since } (\mu, \nu) \text{ is a local star maximum.}
\]
Also, \( \max \{ \psi(e^j, y) \mid y \in Y_0 \} \leq 0, \quad \forall j \in J. \) (Note \( \lambda_j = \infty \)). In addition,
\[
\max \{ \psi(\lambda_i e^i + \mu_j e^j, y) \mid y \in Y_0 \}
\]
\[
= \max \{ \psi(\lambda_i e^i, y) - \psi(-\mu_j e^j, y) \mid y \in Y_0 \}
\]
\[
\leq \max \{ \psi(\lambda_i e^i, y) \mid y \in Y_0 \} - \min \{ \psi(-\mu_j e^j, y) \mid y \in Y_0 \}
\]
\[
\leq \phi(x^*, y^*) + 1 - \phi(x^*, y^*) + 1
\]
\[
= 0
\]
This establishes (3.16).
Algorithm for Bilinear Knapsack Problems

Corollary 6:

\[(3.17) \quad \sum_{i \in I} \frac{x_i}{\lambda_i} \geq 1\]

is a valid cut.

Proof: Let \(z \in \mathbb{R}^m\) be a 0-1 point which is potentially better than the current incumbent. By theorem 5, we have

\[\sum_{i \in I} \frac{x_i}{\lambda_i} + \sum_{j \in J} \frac{y_j}{\Omega_j} \geq 1\]

which implies that (3.17) is valid. \(\Box\)

For those \(i\)'s for which \(x_i(\tilde{z}) \notin x_0\), \(\lambda_i\) can be strictly less than 1. However, a very simple geometric observation leads us to the following theorem:

Theorem 7: Let

\[(3.18) \quad \lambda_i = \max \{1, \tilde{\lambda}_i\}\]

then

\[(3.19) \quad \sum_{i \in I} \frac{x_i}{\lambda_i} \geq 1.\]

is a valid cut.

Proof: It suffices to show that all 0-1 points satisfying \(\sum_{i \in I} \frac{x_i}{\lambda_i} \geq 1\) also satisfies \(\sum_{i \in I} \frac{x_i}{\lambda_i} \geq 1\). Let

\[L = \{i \mid \tilde{\lambda}_i < 1\} \subset I\]

and assume that there exists a 0-1 vector \(z\) such that \(\sum_{i \in \tilde{I}} \frac{x_i}{\lambda_i} \geq 1\) and \(\sum_{i \in \tilde{I}} \frac{x_i}{\lambda_i} < 1\). Then

\[\sum_{i \in \tilde{I}} \frac{x_i}{\lambda_i} + \sum_{i \in \tilde{I}} \frac{x_i}{\lambda_i} \geq 1\]

\[\sum_{i \in \tilde{I}} \frac{x_i}{\lambda_i} + \sum_{i \in \tilde{I}} \frac{x_i}{\lambda_i} < 1\]

which implies that there exists at least one \(i \in L\) for which \(z_i \neq 0\), i.e., \(z_i = 1\).

This in turn implies that \(\sum_{i \in \tilde{I}} \frac{x_i}{\lambda_i} < 1\), a contradiction. \(\Box\)

Remark: \(\sum_{i \in \tilde{I}} \frac{x_i}{\lambda_i} - \sum_{i \in \tilde{I}} \frac{x_i}{\Omega_i} \geq 1\) may not be a valid cut since

\[\max \{\psi(\lambda, z, y) \mid y \in \mathcal{Y}_0\}\]

need not be less than or equal to \(\phi(x^*, y^*) + 1\).

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.
Now we are ready to present the cutting plane algorithm for solving 0-1 bilinear knapsack problems:

Cutting Plane Algorithm \( CBK(X_0', Y_0') \)

Step 0. Let \( X = X_0' \).

Step 1. Execute \( \text{SGO} \left( X, Y_0' \right) \).

Step 2. Compute \( \lambda_i \)'s and \( \mu_j \)'s for \( i \in I \) and \( j \in J \).

\[
X := X \cap \{ z \mid \sum_{i \in I} z_i / \lambda_i \geq 1 \}
\]

Step 3. If \( X \neq \emptyset \) then go to Step 1. Otherwise, stop \((x^*, y^*)\) is an optimal solution of \( (2.1) \).

Theorem 9. \( CBK(X_0', Y_0') \) generates an optimal solution of 0-1 bilinear knapsack problems in finitely many steps.

Proof: At least one integral point of \( X \) is eliminated every time a new cut is adjoined. Hence \( X \) will become empty after finitely many cuts are added to \( X_0' \).

Several comments are in order. First, formula (3.20) may be replaced by:

\[
X := X \cap \{ z \mid \sum_{i \in I} z_i / \lambda_i \geq 1 \}
\]

without invalidating Theorem 9. Second, we would have a deeper cut (i.e., \( \lambda_i \)'s and \( 1 / \mu_j \)'s are larger) if we have a better incumbent \((x^*, y^*)\). Hence it would be advisable to execute procedure \( \text{AMC}(X_0', Y_0'; x^0, y^0) \) by taking several randomly chosen starting points \( x^0, y^0 \), prior to the start of \( CBK(X_0', Y_0') \). Many efficient algorithms have been proposed for 0-1 knapsack problems and this pre-computation would certainly be paid off. Third, we need to solve parametric knapsack problems (3.8), (3.12) to compute the exact value of \( \lambda_i \)'s and \( \mu_j \)'s. However, several efficient approximation procedures are available, two of which will be described in the appendix. Fourth, the algorithm developed in this paper can in principle be adapted to general bilinear programming problems with 0-1 variables:

\[
\begin{align*}
\text{maximize} & \quad \sum_{i=1}^{m} a_i x_i + \sum_{j=1}^{n} d_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_i y_j \\
\text{subject to} & \quad \sum_{i=1}^{m} a_{ri} x_i \leq a_{ro}, \quad r = 1, \ldots, p; \\
& \quad \sum_{j=1}^{n} b_{sj} y_j \leq b_{so}, \quad s = 1, \ldots, q;
\end{align*}
\]

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.
Algorithm for Bilinear Knapsack Problems

\[
x_i \in \{0, 1\}, \quad i = 1, \ldots, m;
\]
\[
y_j \in \{0, 1\}, \quad j = 1, \ldots, n.
\]

Typically our algorithm is expected to work more efficiently on this problem when \( m \) is relatively small (\( m \leq 30 \)) and \( x \) is tightly constrained. It is also desirable that constraints in \( y \)-space to have some nice structure (e.g. assignment constraints) so that subproblem in \( y \)-space can be solved efficiently.

4. Results of Numerical Experiments

We solved about eighty BK's using CBK(\( X_0^0, Y_0^0 \)), in which formula (3.20) was replaced by (3.21). Standard DP algorithm was used to solve \( K_y(x) \), whereas a modified version of RIP-30C code [5] was used to solve \( K_X(y) \). We applied procedure AMC(\( X_0^0, Y_0^0; x^0, y^0 \)) ten times using ten randomly generated starting solutions, prior to the execution of CBK(\( X_0^0, Y_0^0 \)). Also, discrete search scheme given in appendix was used to obtain cut coefficients. Program was coded in FORTRAN IV and computation was done on HITAC M/170. The main purpose of this experiment was (i) to demonstrate the practicality of our approach and (ii) to count the number of cuts to be added before termination to check the effectiveness of cutting planes.

Tables 1 and 2 list some of the results. The density of nonzero coefficients \( c_{ij} \)'s were in the range of 20 to 50 percent. It turned out that optimal solutions have been found for more than 90% of our test problems by applying procedure AMC(\( X_0^0, Y_0^0; y^0 \)) ten times before the start of CBK(\( X_0^0, Y_0^0 \)). Table 3 shows the statistics for 59 test problems of size \((m, n) = (20, 40)\). Total CPU time shows a large standard deviation as is the case for most integer programming problems. Standard deviation of the number of added cuts, on the other hand, is rather small. 42 out of 59 test problems were solved by adding no more than 3 cuts. This shows that our cut is usually very deep. Another code using more efficient branch and bound algorithm for solving \( K_y(x) \) ([1, 11]) and Newton-type procedure to obtain cut coefficient [13] is now under development. Our tentative target is to solve problems of size \( m=40 \). The results of experiments using this code will be reported elsewhere.
Table 1  $m = 10$

<table>
<thead>
<tr>
<th>$n$</th>
<th>number of added cuts</th>
<th>number of times AMC was applied</th>
<th>total CPU time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td>126</td>
<td>5</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>40</td>
<td>2</td>
<td>40</td>
<td>8</td>
</tr>
<tr>
<td>50</td>
<td>2</td>
<td>33</td>
<td>11</td>
</tr>
<tr>
<td>60</td>
<td>2</td>
<td>41</td>
<td>14</td>
</tr>
<tr>
<td>70</td>
<td>3</td>
<td>52</td>
<td>30</td>
</tr>
<tr>
<td>100</td>
<td>3</td>
<td>61</td>
<td>41</td>
</tr>
</tbody>
</table>

Table 2  $m = 20$

<table>
<thead>
<tr>
<th>$n$</th>
<th>number of added cuts</th>
<th>number of times AMC was applied</th>
<th>total CPU time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>2</td>
<td>623</td>
<td>9</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>57</td>
<td>19</td>
</tr>
<tr>
<td>40</td>
<td>4</td>
<td>1152</td>
<td>55</td>
</tr>
<tr>
<td>50</td>
<td>6</td>
<td>46875</td>
<td>584</td>
</tr>
<tr>
<td>60</td>
<td>6</td>
<td>16102</td>
<td>280</td>
</tr>
<tr>
<td>70</td>
<td>8</td>
<td>16842</td>
<td>389</td>
</tr>
<tr>
<td>100</td>
<td>7</td>
<td>24212</td>
<td>451</td>
</tr>
</tbody>
</table>

Also Table 3 shows the statistics for 59 randomly generated $20 \times 40$ test problems.

Table 3

<table>
<thead>
<tr>
<th></th>
<th>number of added cuts</th>
<th>total CPU time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>2.9</td>
<td>64.4</td>
</tr>
<tr>
<td>Minimum</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Maximum</td>
<td>9</td>
<td>265</td>
</tr>
<tr>
<td>Standard deviations</td>
<td>1.6</td>
<td>64.6</td>
</tr>
</tbody>
</table>
Appendix

Approximate Procedures to Obtain Coefficients of a Valid Cut.

One has to compute \( \lambda_{\xi} (\lambda) \) to obtain an exact value of cut coefficients \( \lambda_{\xi} \) in (3.9), which amounts to solving a parametric knapsack problem:

\[
\max \left\{ \sum_{j=1}^{n} (\delta_{j} + \lambda \gamma_{j}) y_{j} \mid \sum_{j=1}^{n} b_{j} y_{j} \leq b_{0}, \right. \\
\left. y_{j} = \{0, 1\}, \quad j = 1, \ldots, m \right\}
\]

Unfortunately, however, there exists no efficient procedure to solve a parametric integer program, so that one needs a procedures to obtain an approximation \( \lambda_{\xi}' \) of \( \lambda_{\xi} \). Note that \( \lambda_{\xi}' \) should be no greater than \( \lambda_{\xi} \) in order that the resulting cut is a valid one.

Procedure 1. (LP Relaxation)

Let

\[
\lambda_{\xi}' = \max \{ \lambda \mid h_{\xi}'(\lambda) \leq \phi(x^{*}, y^{*}) + 1 \}
\]

where

\[
h_{\xi}'(\lambda) = \max \left\{ \sum_{j=1}^{n} (\delta_{j} + \lambda \gamma_{j}) y_{j} \mid \sum_{j=1}^{n} b_{j} y_{j} \leq b_{0}, \quad 0 \leq y_{j} \leq 1, \\
\left. j = 1, \ldots, n \right\} + \gamma_{\xi} \lambda + \phi(\xi, 0)
\]

\( \lambda_{\xi}' \) is expected to give a good approximation of \( \lambda_{\xi} \). Moreover, \( \lambda_{\xi}' \leq \lambda_{\xi} \) since \( h_{\xi}'(\lambda) \geq h_{\xi}(\lambda) \) for all \( \lambda \geq 0 \). What has to be solved here is a parametric linear programming problem instead of a parametric knapsack problem.

The second one is a simple search procedure which uses the convexity of \( h_{\xi} \).

Procedure 2. (Discrete Search)

Step 0. Let \( \alpha > 0, \beta > 1, \lambda = 1 + \alpha \)

Step 1. Compute \( g_{\xi}(\lambda) \)

Step 2. If \( g_{\xi}(\lambda) > \phi(x^{*}, y^{*}) + 1 \), then let \( \alpha = \alpha / 2, \lambda = 1 + \alpha \) and go to Step 1. Otherwise let \( \lambda = \beta \lambda \) and go to Step 3.

Step 3. Compute \( g_{\xi}(\lambda) \).

Step 4. If \( g_{\xi}(\lambda) \leq \phi(x^{*}, y^{*}) + 1 \), then \( \lambda = \beta \lambda \) and go to Step 3. Otherwise let \( \lambda_{\xi}' = \lambda / \beta \) and stop.

\( \lambda_{\xi}' \) computed by this procedure gives an underestimate of \( \lambda_{\xi} \) for \( \xi \in \mathcal{E} \).
Acknowledgements

I wish to thank Messrs. T. Sugiyama and M. Yoshida for their invaluable assistance in numerical experimentation.

References


Hiroshi KONNO: Institute of Information Sciences, University of Tsukuba
Sakuramura, Niiharigun, Ibaraki, 305, Japan.

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.