EIS-TR-79-29


# AN ALGORITHM FOR SOLVING BILINEAR KNAPSACK PROBLEMS

BY

Hiroshi Konno

February 16,1979

INSTITUTE
OF
INFORMATION SCIENCES AND ELECTRONICS

UNIVERSITY OF TSUKUBA

# An Algorithm for Solving Bilinear Knapsack Problems

Hiroshi Konno*

## 1. Introduction

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

$$(BK)\quad \begin{cases} \text{maximize} & \phi(x,y) = \sum_{i=1}^{m} c_i x_i + \sum_{j=1}^{n} d_j y_j + \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_i y_j \\ \text{subject to} & \sum_{i=1}^{m} a_i x_i \leq a_0, \quad x_i = 0 \text{ or } 1, \quad i=1,\cdots,m, \\ & \sum_{j=1}^{n} b_j y_j \leq b_0, \quad y_j = 0 \text{ or } 1, \quad j=1,\cdots,n. \end{cases}$$

where $a_i$'s and $b_j$'s are positive integers and $c_i$'s, $d_j$'s and $c_{ij}$'s are integers. 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):

$$(BL)\quad \begin{cases} \text{maximize} & \sum_{i=1}^{m} c_i x_i + \sum_{j=1}^{n} d_j y_j + \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_i y_j \\ \text{subject to} & \sum_{i=1}^{m} a_{ri} x_i \leq a_r, \quad r=1,\cdots,k\ ; \quad x_i \geq 0, \quad i=1,\cdots,m, \\ & \sum_{j=1}^{n} b_{sj} y_j \leq b_s, \quad s=1,\cdots,\ell\ ; \quad y_j \geq 0, \quad j=1,\cdots,n. \end{cases}$$

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

---

* Associate Professor, Institute of Information Science,
University of Tsukuba, Ibaraki, Japan

[4, 8, 14, 16] for algorithms and to [3, 9, 10, 13, 15] for applications of BL.

It is true that BK can be reformulated as a standard 0-1 integer linear program by introducing mn new 0-1 variables associated with $x_i y_j$'s. But this manipulation will tremendously increase the size of the problem (See Appendix A-2) and it appears that any algorithm which directly exploits the special structure of BK is potentially more efficient.

The main purpose of this paper is to develop a cutting plane algorithm for BK, which parallels the one proposed for BL by the author [8] and more recently by Shetty and Sherali [14].

This algorithm consists of two big procedures. One is to obtain a local maximum which amounts to solving a sequence of 0-1 knapsack problems and 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 best feasible solution identified to date. It will be shown in Section 3 that one has to solve parametric knapsack problems to obtain coefficients of a cut of the above nature. Also, finite convergence of the algorithm (without introducing expensive cuts such as disjunctive face cut [14]) is established by virtue of the discrete nature of the problem. Section 4 will be devoted to a dynamic programming algorithm for BK in which $m = n$, $c_{ij} = 0$, $i \neq j$.

## 2. Algorithm to Obtain a Local Star Maximum

Consider a 0-1 bilinear knapsack problem:

$$\left.\begin{array}{ll} \text{maximize} & \phi(x, y) = \sum_{i=1}^{m} c_i x_i + \sum_{j=1}^{n} d_j y_j + \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_i y_j \\ \text{subject to} & x \in X_0, \quad y \in Y_0. \end{array}\right\} \quad (2.1)$$

where

$$X_0 = \{ x \in R^m \mid \sum_{i=1}^{m} a_i x_i \leq a_0, \quad x_i = 0 \text{ or } 1, \quad i=1, \cdots, m \} \quad (2.2)$$

$$Y_0 = \{ y \in R^n \mid \sum_{j=1}^{n} b_j y_j \leq b_0, \quad y_j = 0 \text{ or } 1, \quad j=1, \cdots, n \} \quad (2.3)$$

It will be assumed throughout that $a_i$'s and $b_j$'s are positive integers and that $c_i$'s, $d_j$'s and $c_{ij}$'s are (not necessarily positive) integers. Also, we will assume that $X_0 \neq \phi$, $Y_0 \neq \phi$ and $(x^*, y^*)$ 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

$$K_X(y) : \text{maximize } \{ \phi(x, y) \mid x \in X \} \quad (2.4)$$

$$K_{Y_0}(x) : \text{maximize } \{ \phi(x, y) \mid y \in Y_0 \} \quad (2.5)$$

The set $X \subset R^m$ is initially taken to be equal to $X_0$, in which case $K_{X_0}(y)$ as well as $K_{Y_0}$ is a 0-1 knapsack problem (For algorithms for 0-1 knapsack problems, see [1, 11, 17]). 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($X, Y_0; y^0$) (Alternate Mountain Climbing)

Step 0. Given $y^0 \in Y_0$, let $k = 1$.

Step 1. Obtain an optimal solution $x^k$ of $K_X(y^{k-1})$ and an optimal solution $y^k$ of $K_{Y_0}(x^k)$.

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 $(\hat{x}, \hat{y}) = (x^k, y^k)$ and go to Step 3.

Step 3. If $\phi(\hat{x}, \hat{y}) > \phi(x^*, y^*)$, then let $(x^*, y^*) = (\hat{x}, \hat{y})$ and go to Step 4.

Step 4. Stop.

The pair of points $(\hat{x}, \hat{y})$ defined above will be called a *locally maximal pair*.

Theorem 1. Procedure AMC$(X, Y_0)$ generates a locally maximal pair in finitely many steps.

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

Once a locally maximal pair is reached, one cannot improve the objective function by fixing the value of either x or y at the current level. In this case, we will switch to a semi-global optimization to be described below.

Let ${}^i x$ be the i-th complement of $x \in X_0$, i.e.,

$$ {}^i x = (x_1, \cdots, x_{i-1}, 1-x_i, x_{i+1}, \cdots, x_m) \tag{2.6} $$

and let

$$ I(x) = \{ i \mid {}^i x \in X \} \tag{2.7} $$

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

$$ \max \{ \phi({}^i\tilde{x}, y) \mid y \in Y_0 \} \leq \phi(\tilde{x}, \tilde{y}) \tag{2.8} $$

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

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

Step 2. If $\max \{ \phi(^i\hat{x}, y) \mid y \in Y_0 \} \leq \phi(\hat{x}, \hat{y})$, $\forall i \in I(\hat{x})$, then stop. Otherwise, go to Step 3.

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

## 3. Cutting Planes from a Local Star Maximum

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

$$\max \{ \phi(x, y) \mid y \in Y_0 \} > \phi(x^*, y^*) \tag{3.1}$$

Such a cutting plane will be called "valid". Note that (3.1) is tantamount to

$$\max \{ \phi(x, y) \mid y \in Y_0 \} \geq \phi(x^*, y^*) + 1 \tag{3.2}$$

since $c_i$'s, $d_j$'s, $c_{ij}$'s together with $x_i$'s and $y_j$'s are all integers.

For simplicity, let

$$z_i = \begin{cases} x_i, & \text{if } \hat{x}_i = 0 \\ 1 - x_i, & \text{if } \hat{x}_i = 1 \end{cases} \qquad i = 1, \cdots, m \tag{3.3}$$

so that $z = 0$ corresponds to $\hat{x}$.

Obviously,

$$\sum_{i=1}^{m} z_i \geq 1 \tag{3.4}$$

is a valid cut since the only point of $X_0$ which is eliminated by this cut is $z = 0$, namely $\hat{x}$. This cut is simple to generate and is sufficient to guarantee finite convergence, but it is shallow and not at all efficient

for practical use. One needs to generate a deeper cut which eliminates more 0-1 points to enhance the efficiency of this algorithm.

For this purpose, let $\psi(z, y)$ be the expression of $\phi(x, y)$ relative to a new set of variables, i.e.,

$$\begin{aligned}\phi(x, y) &= \sum_{i=1}^{m} c_i x_i + \sum_{j=1}^{n} d_j y_j + \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_i y_j \\ &= \sum_{i=1}^{m} \gamma_i z_i + \sum_{j=1}^{n} \delta_j y_j + \sum_{i=1}^{m}\sum_{j=1}^{n} \gamma_{ij} z_i y_j + \phi(\hat{x}, \hat{y}) \\ &= \psi(z, y) \qquad (3.5)\end{aligned}$$

Also, let $g_i(\lambda)$ be the maximum of $\psi$ when z is allowed to move along the edge emanating from $\hat{x}$ in the direction of $z_i$ up to $z_i = \lambda$ and y to vary in $Y_0$, i.e.,

$$g_i(\lambda) = \max \{ \psi(z, y) \mid 0 \le z_i \le \lambda, \quad z_j = 0, \ \forall j \neq i; \quad y \in Y_0 \}$$
$$i = 1, \cdots, m. \qquad (3.6)$$

Theorem 2. $g_i$ is convex on $[0, \infty)$ for all i.

Proof: The maximum of the right hand side of (3.6) is attained either at $z_i = 0$ or at $z_i = \lambda$ since $\psi$ is linear in x. Hence

$$g_i(\lambda) = \max [ \phi(\hat{x}, \hat{y}), h_i(\lambda) ] \qquad (3.7)$$

where

$$h_i(\lambda) = \max \{ \sum_{j=1}^{n} (\delta_j + \gamma_{ij}\lambda) y_j \mid y \in Y_0 \} + \gamma_i \lambda + \phi(\hat{x}, \hat{y}) \qquad (3.8)$$

It is straightforward to see that $h_i$ is convex, whence $g_i$ is convex via (3.7). ||

Given $g_i$, let

$$\hat{\lambda}_i = \max \{ \lambda \mid g_i(\lambda) \le \phi(x^*, y^*) + 1 \} \qquad (3.9)$$

Lemma 3. $\hat{\lambda}_i > 0$ for all i.

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

Theorem 4.

$$\sum_{i=1}^{m} z_i/\hat{\lambda}_i \geq 1 \tag{3.10}$$

is a valid cut.

Proof: Let

$$\Delta(\hat{\lambda}) = \{ z \in R^m \mid \sum_{i=1}^{m} z_i/\hat{\lambda}_i \leq 1, \quad z_i \geq 0, \quad i = 1, \cdots, m \} \tag{3.11}$$

It is sufficient to show

$$\max \{ \psi(z, y) \mid y \in Y_0 \} \leq \phi(x^*, y^*) + 1, \qquad \forall z \in \Delta(\hat{\lambda})$$

To show this, let us fix $z \in \Delta(\hat{\lambda})$. Then z can be expressed as $z = \sum_{i=0}^{m} \theta_i \Lambda_i$ where $\Lambda_i$, $i = 0, 1, \cdots, m$ are vertices of $\Delta(\hat{\lambda})$ and $\sum_{i=0}^{m} \theta_i = 1$, $\theta_i \geq 0$, $i = 0, 1, \cdots, m$, so that

$$\psi(z, y) = \psi(\sum_{i=0}^{m} \theta_i \Lambda_i, y) = \sum_{i=0}^{m} \theta_i \psi(\Lambda_i, y)$$

Therefore,

$$\begin{aligned} \max \{ \psi(z, y) \mid y \in Y_0 \} &\leq \sum_{i=0}^{m} \theta_i \max \{ \psi(\Lambda_i, y) \mid y - Y_0 \} \\ &\leq \sum_{i=0}^{m} \theta_i \{ \phi(x^*, y^*) + 1 \} \\ &= \phi(x^*, y^*) + 1. \end{aligned}$$ ||

The next theorem shows that the cut (3.10) is deeper than the cut (3.4) in the derection of $z_i$ for $i \in I(\hat{x})$.

Theorem 5. $\hat{\lambda}_i > 1$ for all i such that ${}^i\hat{x} \in X_0$.

Proof: It is sufficient to show that $g_i(1) \leq \phi(x^*, y^*) + 1$ since $g_i$ is convex and $g_i(0) = \phi(\hat{x}, \hat{y})$. If ${}^i\hat{x} \in X$, then

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

by the definition of local star maximality. Also if ${}^ix \in X_0 - X$, then

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

since $^i\hat{x}$ has been cut off by a valid cut adjoined earlier. ||

The next theorem gives an easy sufficient condition for global optimality.

Theorem 6. If $\sum_{i=1}^{m} 1/\hat{\lambda}_i \leq 1$, then $(x^*, y^*)$ is optimal to (2.1).

Proof: Every 0-1 vector z is contained in $\Delta(\hat{\lambda})$. ||

If $\hat{\lambda}_i = \infty$ (which is typically the case when $\delta_i \leq 0$ and $\gamma_{ij} \leq 0$, $\forall j$), one can generate an even deeper cut by using negative edge extensions [12, 13]. Let

$$I = \{ i \mid \hat{\lambda}_i < \infty \} \tag{3.12}$$

$$J = \{ i \mid \hat{\lambda}_i = \infty \} \tag{3.13}$$

and for $i \in J$, let

$$G_i(\mu) = \min \{ \psi(z, y) \mid -\mu \leq z_i \leq 0,\ z_j = 0, \forall j \neq i;\ y \in Y_0 \} \tag{3.14}$$

$$\hat{\mu}_i = \max \{ \mu \mid G_i(\mu) \leq \phi(x^*, y^*) + 1 \} \tag{3.15}$$

Lemma 7. $\hat{\mu}_i > 0$ 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 8.

$$\sum_{i \in I} z_i/\hat{\lambda}_i - \sum_{i \in J} z_i/\hat{\mu}_i \geq 1 \tag{3.16}$$

is a valid cut.

Proof: Let

$$Z = \{ z \in R^m \mid \sum_{i \in I} z_i/\hat{\lambda}_i - \sum_{i \in J} z_i/\hat{\mu}_i \leq 1,\ z_i \geq 0,\ i = 1, \cdots, m \} \tag{3.17}$$

One needs to show

$$\max \{ \psi(z, y) \mid y \in Y_0 \} \leq \phi(x^*, y^*) + 1, \quad \forall z \in Z \tag{3.18}$$

It is easy to see that the extreme points of Z are given by

$$z^i = \hat{\lambda}_i e^i, \qquad i \in I$$

and the extreme directions of Z are given by

(i) $e^j$, $\quad j \in J$.

(ii) $\hat{\mu}_j e^j + \hat{\lambda}_i e^i$, $\quad j \in J, \quad i \in I$.

where $e^i$ and $e^j$ are the i-th and j-th unit vector, respectively. Hence $z \in Z$ can be expressed as

$$z = \sum_{i\in I}\theta_i z^i + \sum_{j\in J}\alpha_j e^j + \sum_{i\in I}\sum_{j\in J}\alpha_{ij}(\hat{\mu}_j e^j + \hat{\lambda}_i e^i)$$

where $\sum_{i\in I}\theta_i = 1$, $\theta_i \geq 0$, $\forall i \in I$; $\alpha_j \geq 0$, $\forall j \in J$; $\alpha_{ij} \geq 0$, $\forall i \in I$, $\forall j \in J$ so that

$$\psi(z, y) = \sum_{i\in I}\theta_i\psi(z^i, y) + \sum_{j\in J}\alpha_j\psi(e^j, y) + \sum_{i\in I}\sum_{j\in J}\alpha_{ij}\psi(\hat{\mu}_j e^j + \hat{\lambda}_i e^i, y)$$

Hence

$$\max\{\psi(z, y) \mid y\in Y_0\} \leq \sum_{i\in I}\theta_i \max\{\psi(z^i, y) \mid y\in Y_0\} + \sum_{j\in J}\alpha_j \max\{\psi(e^j, y) \mid y\in Y_0\}$$
$$+ \sum_{i\in I}\sum_{j\in J}\alpha_{ij}\max\{\psi(\hat{\mu}_j e^j + \hat{\lambda}_i e^i, y) \mid y\in Y_0\}$$

The first term of the right hand side is less than $\phi(x^*, y^*)+1$ since $\max\{\psi(z^i, y) \mid y \in Y_0\} \leq \phi(x^*, y^*) + 1$, $\forall i \in I$. The second term is less than 0 since $\max\{\psi(e^j, y) \mid y \in Y_0\} \leq 0$, $\forall j \in J$. (Note $\hat{\lambda}_j = \infty$). Also

$$\max\{\psi(\hat{\lambda}_i e^i + \hat{\mu}_j e^j, y) \mid y \in Y_0\}$$
$$= \max\{\psi(\hat{\lambda}_i e^i, y) - \psi(-\hat{\mu}_j e^j, y) \mid y \in Y_0\}$$
$$\leq \max\{\psi(\hat{\lambda}_i e^i, y) \mid y \in Y_0\} - \min\{\psi(-\hat{\mu}_j e^j, y) \mid y \in Y_0\}$$
$$\leq \{\phi(x^*, y^*) + 1\} - \{\phi(x^*, y^*) + 1\} = 0$$

This establishes (3.18). ||

Now we are ready to present the cutting plane algorithm for solving 0-1 bilinear knapsack problems:

Cutting Plane Algorithm $\quad CBK(X_0, Y_0)$

Step 0. Let $X = X_0$.

Step 1. Execute $SGO(X, Y_0)$.

Step 2. Compute $\hat{\lambda}_i$'s and $\hat{\mu}_j$'s for $i \in I$ and $j \in J$.

$$X := X \cap \{ z \mid \sum_{i \in I} z_i/\hat{\lambda}_i - \sum_{j \in J} z_i/\hat{\mu}_i \geq 1 \}$$

Step 3. If $X \neq \phi$ 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 point of X is eliminated every time a new cut is adjoined. Therefore, the set X will become empty after finitely many cuts are added to $X_0$. ||

Several comments are in order. First, we would get a deeper cut (i.e., $\hat{\lambda}_i$'s and $\hat{\mu}_j$'s are larger) if we have a better incumbent $(x^*, y^*)$. Hence it would be advisable to execute procedure $AMC(X_0, Y_0; y^0)$ by taking several randomly chosen starting points $y^0$, prior to start $CBK(X_0, Y_0)$. Many efficient algorithms have been proposed for 0-1 knapsack problems and this pre-computation would certainly be paid off. Second, we need to solve parametric knapsack problems (3.8) to obtain an exact value of $\hat{\lambda}_i$, which is not at all an easy task. However, several efficient approximation procedures can be constructed, which will be discussed in the appendix. Third, the algorithm developed in this paper can in principle be adapted to general bilinear programming problems with 0-1 variables. It is, however, desirable that the constraints on y have a special structure so that one can generate a valid cut without too much computation. The most important problems in this category would be bilinear assignment problems, which will be discussed in a subsequent paper.

## 4. Dynamic Programming Algorithms for a Special Class of 0-1 Bilinear Knapsack Problems

Let us consider here a 0-1 bilinear knapsack problem in which $m = n$ and $c_{ij} = 0$ for all $i \neq j$;

$$\left.\begin{array}{lll} \text{maximize} & \Phi(x, y) = \sum_{j=1}^{n}(c_j x_j + d_j y_j + e_j x_j y_j) & \\ \text{subject to} & \sum_{j=1}^{n} a_j x_j \leq a_0, & x_j = 0 \text{ or } 1, \\ & \sum_{j=1}^{n} b_j y_j \leq b_0, & y_j = 0 \text{ or } 1. \end{array}\right\} \quad (4.1)$$

where, as before, $a_j$'s and $b_j$'s are positive integers and $c_j$, $d_j$, $e_j$'s are all integers.

For $s = 0, 1, \cdots, a_0$ and $t = 0, 1, \cdots, b_0$, let

$$f_k(s, t) = \max \{ \sum_{j=1}^{k}(c_j x_j + d_j y_j + e_j x_j y_j) \mid \sum_{j=1}^{k} a_j x_j \leq s,\ x_j = 0 \text{ or } 1,$$
$$j = 1, \cdots, k;\ \sum_{j=1}^{k} b_j y_j \leq t,\ y_j = 0 \text{ or } 1,\ j = 1, \cdots, k\}$$
$$k = 1, \cdots, n. \quad (4.2)$$

Obviously $f_n(a_0, b_0)$ gives the maximum of $\Phi(x, y)$ in (4.1).

Theorem 10 $f_k(s, t)$ satisfies the following recursion

$$f_k(s, t) = \max \{ f_{k-1}(s-a_k, t-b_k) + c_k + d_k + e_k,\ f_{k-1}(s-a_k, t) + c_k,$$
$$f_{k-1}(s, t-b_k) + d_k,\ f_{k-1}(s, t) \} \quad (4.3)$$

with boundary conditions

$$\left.\begin{array}{ll} f_k(s, t) = 0 & \text{for } s \leq 0 \text{ or } t \leq 0, \ \forall k, \\ f_0(s, t) = 0 & \text{for all } s, t. \end{array}\right\} \quad (4.4)$$

Proof: Straightforward and will be omitted. ||

If, in addition, $c_j = d_j = 0$ for all $j$ in (4.1), a more efficient recursion can be constructed. Let us consider now

$$\left.\begin{array}{ll} \text{maximize} & \psi(x, y) = \sum_{j=1}^{n} e_j x_j y_j \\ \text{subject to} & \sum_{j=1}^{n} a_j x_j \le a_0, \quad x_j = 0 \text{ or } 1, \quad j=1, \cdots, n, \\ & \sum_{j=1}^{n} b_j y_j \le b_0, \quad y_j = 0 \text{ or } 1, \quad j=1, \cdots, n. \end{array}\right\} \quad (4.5)$$

Associated with this problem is a standard 2-dimensional 0-1 knapsack problem:

$$\left.\begin{array}{ll} \text{maximize} & \theta(u) = \sum_{j=1}^{n} e_j u_j \\ \text{subject to} & \sum_{j=1}^{n} a_j u_j \le a_0, \\ & \sum_{j=1}^{n} b_j u_j \le b_0, \quad u_j = 0 \text{ or } 1, \quad j=1, \cdots, n. \end{array}\right\} \quad (4.6)$$

Theorem 11 If $u^*$ is optimal to (4.6), then $(x, y) = (u^*, u^*)$ is optimal to (4.5).

Proof: Obviously, both problems (4.5) and (4.6) have optimal solutions. Let $(x^*, y^*)$ be optimal to (4.5) and assume without loss of generality that $x_j^* y_j^* = 1$ for $j = 1$ to $k$ and $x_j^* y_j^* = 0$, otherwise, so that $\psi^* = \sum_{j=1}^{n} e_j x_j^* y_j^* = \sum_{j=1}^{k} e_j$. $\theta(u^*) \ge \psi^*$ since $u_j = 1$, $j = 1, \cdots, k$, $u_j = 0$, $j = k+1, \cdots, n$ is feasible to (4.6). Also, $\psi^* \ge \sum_{j=1}^{n} e_j u_j^* u_j^* = \sum_{j=1}^{n} e_j u_j^* = \theta(u^*)$ since $(x, y) = (u^*, u^*)$ is feasible to (4.5). This establishes $\theta(u^*) = \psi^*$. Hence $\psi(u^*, u^*) = \theta(u^*) = \psi^*$, namely, $(u^*, u^*)$ is optimal to (4.5). ||

By virtue of this theorem, the recursion in (4.3) simplifies as follows:

$$g_k(s, t) = \max \{ g_{k-1}(s, t), \; g_{k-1}(s-a_k, t-b_k) + e_k \} \quad (4.3')$$

with boundary conditions:

$$\begin{array}{ll} g_k(s, t) = 0, & s \le 0, \; t \le 0, \; \forall k. \\ g_0(s, t) = 0 & \text{for all } s \text{ and } t. \end{array} \quad (4.4')$$

As before, $g_n(a_0, b_0)$ gives the optimal value $\psi^*$ of the objective function

$\psi$ in (4.5). (4.3') can also be derived from (4.3) by considering that $f_{k-1}(s-a_k, t) + c_k$ and $f_{k-1}(s, t-b_k) + d_k$ are no better than $f_{k-1}(s, t)$ when $c_k = d_k = 0$.

## Appendix

A-1. Approximate Procedures to Obtain Coefficients of a Valid Cut.

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

$$\text{maximize} \{ \sum_{j=1}^{n} (\delta_j + \lambda\gamma_{ij}) y_j \mid \sum_{j=1}^{n} b_j y_j \leq b_0,\ y_j = 0 \text{ or } 1,\ j = 1, \cdots, m \} \quad (A.1)$$

Unfortunately, however, there exists no efficient procedure to solve a parametric integer program, so that one needs a procedures to obtain an approximation $\bar{\lambda}_i$ of $\hat{\lambda}_i$. Note that $\bar{\lambda}_i$ should be no greater than $\hat{\lambda}_i$ in order that the resulting cut is a valid one.

Procedure 1. (LP Relaxation)

Let

$$\bar{\lambda}_i = \max \{ \lambda \mid \bar{h}_i(\lambda) \leq \phi(x^*, y^*) + 1 \} \quad (A.2)$$

where

$$\bar{h}_i(\lambda) = \max \{ \sum_{j=1}^{n} (\delta_j + \lambda\gamma_{ij}) y_j \mid \sum_{j=1}^{n} b_j y_j \leq b_0,\ 0 \leq y_j \leq 1,$$
$$j = 1, \cdots, n \} + \gamma_i \lambda + \phi(\hat{x}, \hat{y}) \quad (A.3)$$

$\bar{\lambda}_i$ is expected to give a good approximation of $\hat{\lambda}_i$. Moreover, $\bar{\lambda}_i \leq \hat{\lambda}_i$ since $\bar{h}_i(\lambda) \geq h_i(\lambda)$ 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 $h_i$.

Procedure 2. (Discrete Search)

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

Step 1. Compute $g_i(\lambda)$.

Step 2. If $g_i(\lambda) > \phi(x^*, y^*) + 1$, 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 $g_i(\lambda)$.

Step 4. If $g_i(\lambda) \leq \phi(x^*, y^*) + 1$, then $\lambda = \beta\lambda$ and go to Step 3. Otherwise let $\bar{\lambda}_i = \lambda/\beta$ and stop.

1)

$\bar{\lambda}_i$ computed by this procedure gives an underestimate of $\hat{\lambda}_i$ for $i \in I(\hat{x})$. Similar procedure can be constructed for $i \notin I(\hat{x})$ as well, but will be omitted here.

## A-2. Alternative Formulation of Bilinear Knapsack Problems as Standard 0-1 Integer Programming Problems

The purpose of this appendix is to show that it is possible to

2) formulate BK as an integer linear program by introducing new 0-1 variables:

$$\left.\begin{array}{ll} \text{maximize} & \sum_{i=1}^{m} c_i x_i + \sum_{j=1}^{n} d_j y_j + \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} u_{ij} \\ \text{subject to} & x \in X_0, \quad y \in Y_0, \\ & \left.\begin{array}{l} u_{ij} \leq \frac{1}{2}(x_i + y_j) \\ u_{ij} \geq \frac{1}{2}(x_i + y_j - 1) \\ u_{ij} = 0 \text{ or } 1, \end{array}\right\} \quad \begin{array}{l} i=1, \cdots, m, \\ j=1, \cdots, n \end{array} \end{array}\right\} \quad (A.4)$$

3)

Note that $u_{ij} = 1$ if and only if $x_i = y_j = 1$ by virtue of the constraints on $u_{ij}$'s. This problem has $mn + m + n$ variables and $2mn + m + n$ constraints, which are much bigger than those of BK.

If, however, $c_{ij} \geq 0$ for all i and j, one can eliminate almost one half of the constraints in (A.4) as follows:

$$\left.\begin{array}{ll} \text{maximize} & \sum_{i=1}^{m} c_i x_i + \sum_{j=1}^{n} d_j y_j + \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} u_{ij} \\ \text{subject to} & x \in X_0, \quad y \in Y_0, \end{array}\right\} \quad (A.5)$$

$$\left.\begin{aligned} u_{ij} &\leq \tfrac{1}{2}(x_i + y_j) && i=1, \cdots, m, \\ u_{ij} &= 0 \text{ or } 1 && j=1, \cdots, n. \end{aligned}\right\} \qquad (A.5)$$

It is easy to see that $u_{ij}$ can be equal to 1 in (A.5) only if $x_i = y_j = 1$. Also, if $x_i = y_j = 1$ at the optimum, then $u_{ij}$ should be equal to 1 since $c_{ij} \geq 0$. This establishes the equivalence of (A.5) and (A.4) when $c_{ij} \geq 0$ for all i and j.

References


1. Balas, E. and E. Zemel, "Solving Large Zero-One Knapsack Problems", MSRR 408, GSIA, Carnegie-Mellon University, (1977).
2. Balas, E., "Disjunctive Programming", MSRR 415, GSIA, Carnegie-Mellon University, (1977).
3. Frieze, A. M., "A Bilinear Programming Formulation of the 3-dimensional Assignment Problem", *Mathematical Programming 7* (1974), pp. 376-379.
4. Falk, J. E., "A Linear Max-Min Problem", *Mathematical Programming 5* (1973), pp. 169-188.
5. Gallo, G. and A. Ülkücü, "Bilinear Programming: An Exact Algorithm", *Mathematical Programming 12* (1977), pp. 173-194.
6. Geoffrion, A. M., "An Improved Implicit Enumeration Approach for Integer Programming", *Operations Research 17* (1969), pp. 437-454.
7. Gilmore, P. C. and R. E. Gomory, "Multistage Cutting Stock Problems of Two and More Dimensions", *Operations Research 13* (1965), pp. 94-120.
8. Konno, H., "A Cutting Plane Algorithm for Solving Bilinear Programs", *Mathematical Programming 11* (1976), pp. 14-27.
9. Konno, H., "Bilinear Programming PART II: Applications of Bilinear Programming", Technical Report 71-10, Dept. of OR, Stanford University, (1971).
10. Konno, H., "On the Applications of Bilinear Programming", EIS-TR-79-1, Institute of Information Science, University of Tsukuba, Japan 1979 (Submitted to the J. of Optimization Theory and Applications).
11. Nauss, R. M., "An Efficient Algorithm for the 0-1 Knapsack Problem", *Management Science 23* (1976), pp. 27-31.
12. Owen, G., "Cutting Planes for Programs with Disjunctive Constraints", *J. of Optimization Theory and Applications 11* (1973), pp. 49-55.
13. Shetty, C. M. and H. D. Sherali, "Rectilinear Distance Location-Allocation Problem: A Simplex Based Algorithm", *Proceedings of the International Symposium on Extreme Methods and Systems Analysis,* Springer, 1978.
14. Shetty, C. M. and H. D. Sherali, "A Finitely Convergent Algorithm for Bilinear Programming Problems Using Polar Cuts and Disjunctive-Face Cuts", to appear in *Mathematical Programming*.
15. Vaish, H. and C. M. Shetty, "The Bilinear Programming Problem", *Naval Research Logistics Quarterly 23* (1976), pp. 303-309.
16. Vaish, H. and C. M. Shetty, "A Cutting Plane Algorithm for the Bilinear Programming Problems", *Naval Research Logistics Quarterly 24* (1977), pp. 83-94.
17. Zoltners, A. A., "A Direct Descent Binary Knapsack Algorithm", *J. ACM 25* (1978), pp. 304-311.

INSTITUTE OF ELECTRONICS AND INFORMATION SCIENCE
UNIVERSITY OF TSUKUBA
SAKURA-MURA, NIIHARI-GUN, IBARAKI JAPAN

| REPORT DOCUMENTATION PAGE | REPORT NUMBER |
|---|---|

TITLE

An Algorithm for Solving Bilinear Knapsack Problems

AUTHOR(s)

Hiroshi Konno

| REPORT DATE | NUMBER OF PAGES |
|---|---|
| February 16, 1979 | 17 |
| MAIN CATEGORY | CR CATEGORIES |
| Mathematical Programming | 5. 4 |

KEY WORDS

bilinear programming, knapsack problem, 0-1 integer program, cutting plane method, dynamic programming

ABSTRACT

This paper introduces 0-1 bilinear knapsack problems (BK) and proposes a finitely convergent cutting plane algorithm which parallels the one proposed for bilinear linear programming problem by the present author.

This algorithm consists of two big procedures. One is to obtain a local maximum, which amounts to solving a sequence of 0-1 knapsack problems and 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 that one has to solve parametric knapsack problems to obtain coefficients of a cut. Also, finite convergence of the algorithm is established by virtue of the discrete nature of the problem. It will be shown, in addition, that a dynamic programming algorithm can be constructed for a BK with a special structure.

SUPPLEMENTARY NOTES