Journal of the Operations Research Society of Japan
Vol. 38, No. 2, June 1995

# OUTER APPROXIMATION ALGORITHMS FOR LOWER RANK BILINEAR PROGRAMMING PROBLEMS

Yasutoshi Yajima Hiroshi Konno
*Tokyo Institute of Technology*

(Received August 11, 1993; Revised April 13, 1994)

*Abstract* Two outer approximation algorithms for lower rank bilinear programming problems are developed. The first algorithm can generate an $\epsilon$-optimal solution rather efficiently when the rank of the objective function is less than five. The second algorithm is exact and finitely convergent, yet with slower convergent property, compared to the first.

## 1. Introduction

Among the rapidly developing field of global optimization, minimization of a nonconvex quadratic function:

(1.1) $f(x) = c^tx + \frac{1}{2}x^tQx$

over a polytope has been the subject of most extensive research. In particular, concave quadratic programming problems (CQP) and bilinear programming problems (BLP) have been studied by many researchers because they have a number of nice applications [2, 3, 7, 13] in production and inventory control problems, location-allocation problems, computational geometry, portfolio selection, combinatorial optimization, etc. Also, several algorithms [1, 5, 6, 12, 19] have been proposed for these problems using the fundamental property that they have optimal solutions among extreme points [18]. Unfortunately, however, general CQP's and BLP's with over one hundred variables are still beyond reach of general purpose algorithm from the practical point of view [16].

Very recently, a new and promising approach has been proposed for lower rank nonconvex quadratic programming problems [8, 11]. By lower rank problems, we mean problems in which the rank of $Q$ is at most 4 or 5.

One such example is linear multiplicative programming problems:

(1.2) $\text{minimize}\{c_0^tx + c_1^tx \cdot c_2^tx \mid x \in X\}$,

where $X$ is a polytope. This problem, though simple looking, is known to be NP complete [15]. On the other hand, we showed in [11] that a variant of primal-dual parametric simplex algorithm can generate a global minimum of this function in more or less the same amount of computational time needed to solve a linear programming problem of the same size. Also, it has been shown [9] that this algorithm has the "average polynomial order property" under the probability distribution of the data commonly assumed in the probabilistic analysis of the simplex method for linear programs [17].

Another example is rank two bilinear programming problems:

(1.3) $\text{minimize}\{c_0^tx + d_0^ty + c_1^tx \cdot d_1^ty + c_2^tx \cdot d_2^ty \mid x \in X,\ y \in Y\}$,

where $X$ and $Y$ are polytopes. We showed in [20] that a two parameter primal-dual simplex algorithm can solve this problem in about $4 \sim 5$ times computational time required to solve an associated linear programming problem. In short, we can solve large scale nonconvex quadratic programming problems (1.2) and (1.3) with a few hundreds variables or even more. Yet another class of problems we have solved is rank two nonconvex quadratic programming problems:

(1.4) $\quad \text{minimize}\{c_0^t x + c_1^t x \cdot c_2^t x + c_3^t x \cdot c_4^t x \mid x \in X\}.$

It has been shown that an alternative parametrization technique combined with outer approximation method works well for this class of problems.

The purpose of this paper is to develop a practical algorithm for solving up to rank five bilinear programming problems for which there are yet no efficient algorithms. The reason why we are particularly interested in this class of problems are that (1) general concave quadratic programming problems can be converted to equivalent bilinear programming problems [5], and that (2) approximate optimal solutions of general concave quadratic programming problems and bilinear programming problems can be obtained by solving associated lower rank problems in the following way.

Let $Q$ be a real symmetric matrix, it is known that there exists an orthonormal matrix

$$(1.5) \qquad P = \begin{bmatrix} g_1^t \\ \vdots \\ g_n^t \end{bmatrix},$$

such that

$$(1.6) \qquad \begin{aligned} x^t Q x &= x^t P^t \Lambda P x \\ &= \sum_{i=1}^{n} \lambda_i (g^t x)^2, \end{aligned}$$

where $\Lambda$ is a diagonal matrix whose elements are eigenvalues $\lambda_i$, $i = 1, \dots, n$ of $Q$. Let us assume that

$$|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|.$$

Then we can obtain an approximation of the quadratic function by a lower rank quadratic function by taking the first $p$ terms of (1.6). In fact, this approach turns out to be very efficient for certain class of real world problems including portfolio analysis [10].

In Section 2 we will introduce a parametric representation of rank $p$ bilinear programming problem. In Section 3 we develop a practical algorithm which generates an $\epsilon-$ optimal solution. Also some results of numerical tests will be reported. Finally in Section 4 we will propose an exact and finitely convergent outer approximate algorithm for the same problem.

## 2. Rank $p$ Bilinear Programming Problems

Let us consider a rank $p$ bilinear programming problem:

$$(2.7) \qquad \left| \begin{array}{ll} \text{minimize} & c_0^t x + d_0^t y + \sum_{i=1}^{p} c_i^t x \cdot d_i^t y \\ \text{subject to} & x \in X, \ y \in Y, \end{array} \right.$$

where $c_i \in R^{n_1}$, $d_i \in R^{n_2}$ $(i = 0, 1, \dots, p)$ and

(2.8) $\quad X = \{x \in R^{n_1} \mid A_1 x \leq b_1\},$

(2.9) $\quad Y = \{y \in R^{n_2} \mid A_2 y \leq b_2\},$

where $A_1 \in R^{m_1 \times n_1}$, $b_1 \in R^{m_1}$, $A_2 \in R^{m_2 \times n_2}$, $b_2 \in R^{m_2}$. We assume that $c_i$'s and $d_i$'s are linearly independent and that $X, Y$ are non-empty and bounded polytopes.

To solve (2.7), we will introduce $p + 1$ auxiliary variables:

(2.10) $\quad \xi_i = d_i^t y, \ i = 0, 1, \dots, p,$

and define the problem:

$$(2.11) \qquad \left| \begin{array}{ll} \text{minimize} & c_0^t x + \xi_0 + \sum_{i=1}^{p} \xi_i \cdot c_i^t x \\ \text{subject to} & x \in X, \ y \in Y, \\ & \xi = Dy, \end{array} \right.$$

where

(2.12) $D = \begin{bmatrix} d_0^t \\ \vdots \\ d_p^t \end{bmatrix}$.

The following relationship between (2.7) and (2.11) is obvious.

**Theorem 2.1** *Let $(x^*, y^*, \xi_0^*, \xi_1^*, \ldots, \xi_p^*)$ be an optimal solution of (2.11), then $(x^*, y^*)$ is an optimal solution of (2.7).*

Let

(2.13) $G = \{\xi \in R^{p+1} \mid \xi = Dy,\ A_2 y \le b_2\}$,

and let us define a subproblem:

$$(2.14)\quad \left|\begin{array}{ll} \text{minimize} & f(x;\xi) \equiv c_0^t x + \xi_0 + \sum_{i=1}^{p} \xi_i \cdot c_i^t x \\ \text{subject to} & x \in X. \end{array}\right.$$

Also, let $\xi \in G$ and

(2.15) $F(\xi) = \text{minimize}\{f(x;\xi) \mid x \in X\}$.

**Lemma 2.2** *$F(\xi)$ is a concave function.*

**Proof** Let $\lambda \in (0,1)$ and $\xi^1, \xi^2 \in G$.

$$\begin{aligned} & F(\lambda\xi^1 + (1-\lambda)\xi^2) \\ = \ & \min\{\lambda(c_0^t x + \xi_0^1 + \sum_{i=1}^{p} \xi_i^1 c_i^t x) + (1-\lambda)(c_0^t x + \xi_0^2 + \sum_{i=1}^{p} \xi_i^2 c_i^t x) \mid x \in X\}, \\ \ge \ & \lambda \min\{(c_0^t x + \xi_0^1 + \sum_{i=1}^{p} \xi_i^1 c_i^t x) \mid x \in X\} \\ & +(1-\lambda)\min\{(c_0^t x + \xi_0^2 + \sum_{i=1}^{p} \xi_i^2 c_i^t x) \mid x \in X\}, \\ = \ & \lambda F(\xi^1) + (1-\lambda) F(\xi^2). \end{aligned}$$

□

Thus problem (2.7) is reduced to the following concave minimization problem with $p+1$ variables.

$$(2.16)\quad \left|\begin{array}{ll} \text{minimize} & F(\xi) \\ \text{subject to} & \xi \in G. \end{array}\right.$$

It is well known that a global minimum of (2.16) is attained at one of the vertices of $G$.

## 3. Algorithm for $\epsilon$–Optimal Solution

We will apply an outer approximation algorithm for solving the concave minimization problem (2.16).

For this purpose, let

$$\underline{\xi}_i = \min\{\xi_i \mid (\xi_0, \xi_1, \ldots, \xi_p)^t \in G\},\quad i = 0, 1, \ldots, p,$$
$$\overline{\xi}_i = \max\{\xi_i \mid (\xi_0, \xi_1, \ldots, \xi_p)^t \in G\},\quad i = 0, 1, \ldots, p,$$

which can be obtained by solving $2(p+1)$ linear programming problems: Note that $\underline{\xi}_i$'s and $\overline{\xi}_i$'s are finite since $Y$ is bounded. Thus $G$ is contained in a hypercube:

$$G^0 = \{\xi \in R^{p+1} \mid \underline{\xi}_i \le \xi_i \le \overline{\xi}_i,\ i = 0, 1, \ldots, p\}.$$

Thus we define the initial relaxation problem:

$$(3.17)\quad (BP)^0 \left|\begin{array}{ll} \text{minimize} & F(\xi) \\ \text{subject to} & \xi \in G^0, \end{array}\right.$$

whose optimal solution is attained at either one of the $2^{p+1}$ vertices of the hypercube $G^0$.

Let us consider the $k$th relaxation problem:

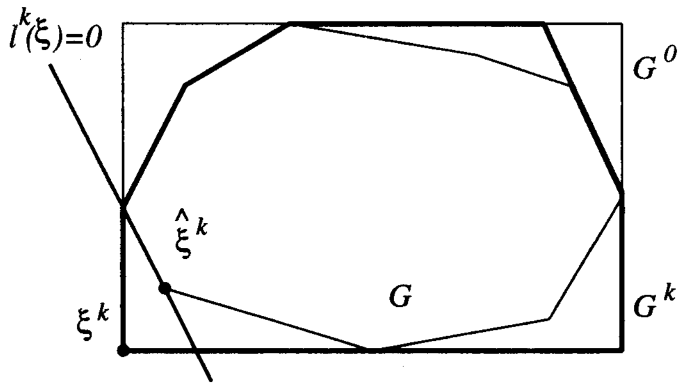

Figure 3.1: Figure of Cut

$$(3.18)\quad (BP)^k \left|\begin{array}{ll} \text{minimize} & F(\xi) \\ \text{subject to} & \xi \in G^k, \end{array}\right.$$

where $G^0 \supset G^k \supseteq G$ and let $\xi^k$ be an extreme optimal solution of $(BP)^k$. If $\xi^k$ does not belong to $G$, we define the following affine function

$$(3.19)\quad l^k(\xi) = \sum_{i=0}^{p}(\hat{\xi}_i^k - \xi_i^k)(\xi_i - \hat{\xi}_i^k)$$

where

$$(3.20)\quad \hat{\xi}^k = \operatorname{argmin}\{\| \xi - \xi^k \| \mid \xi \in G\}.$$

The hyperplane $l^k(\xi) = 0$ supports $G$ at $\hat{\xi}^k$, which can be obtained by solving the following convex quadratic problem:

$$(3.21)\quad \left|\begin{array}{ll} \text{minimize} & \sum_{i=0}^{p}(\xi_i - \xi_i^k)^2 \\ \text{subject to} & y \in Y, \\ & \xi_i = d_i^t y, \ \ i = 0, \ldots, p. \end{array}\right.$$

It is easy to see that if $\xi^k \notin G$ then

$$l^k(\xi^k) < 0, \ \ l^k(\xi) \geq 0, \ {}^\forall\xi \in G.$$

Whence we can construct $G^{k+1}$ by adding a cut

$$(3.22)\quad L^k = \{\xi \in R^{p+1} \mid l^k(\xi) \geq 0\}$$

which cuts off $\xi^k$.

**Algorithm A**

```
Step1 Let $k = 0$.
Step2 Compute an optimal solution $\xi^k$ of the relaxation problem (BP)$^k$.
Step3 Construct the cut $L^k$ by (3.19) and (3.20).
Step4 If $\| \xi^k - \hat{\xi}^k \| < \epsilon$ then stop. Otherwise, let
          $G^{k+1} = G^k \cap L^k, \ \ k = k + 1,$
      and return to Step 2.
```

□

**Theorem 3.3** *The algorithm* **A** *generates a sequence of points $\xi^k$ which converge to an $\epsilon-$ optimal solution $\xi^*$ of (2.16).*

**Proof** See [4] (Theorem II.3). □

Note that even if $\xi^k \notin G$, the associated $(x^k, y^k)$ is always a feasible solution of the original problem.

Table 3.1: Results of (3.23) when $p = 4$ and $\epsilon = 10^{-5}$

| $m$ | 80 | 80 | 100 | 100 | 120 | 120 |
|---|---|---|---|---|---|---|
| $n$ | 60 | 100 | 100 | 120 | 120 | 140 |
| CPU time (in seconds) | | | | | | |
| Av. | 144.71 | 446.34 | 574.61 | 824.16 | 1604.06 | 1931.06 |
| S.d. | (30.43) | (278.09) | (144.49) | (269.93) | (838.22) | (712.34) |
| # of cuts | | | | | | |
| Av. | 27.1 | 33.1 | 31.3 | 32.4 | 32.4 | 33.4 |
| S.d. | ( 3.11) | ( 9.54) | ( 4.58) | ( 4.63) | ( 7.81) | ( 8.10) |
| # of vertices | | | | | | |
| Av. | 255.4 | 350.0 | 322.8 | 329.8 | 324.0 | 334.6 |
| S.d. | (46.04) | (161.77) | (76.96) | (70.33) | (102.98) | (127.26) |

Table 3.2: Results of (3.23) when $p = 5$ and $\epsilon = 10^{-5}$

| $m$ | 50 | 50 | 80 | 80 | 100 | 100 |
|---|---|---|---|---|---|---|
| $n$ | 40 | 60 | 60 | 100 | 80 | 100 |
| CPU time (in seconds) | | | | | | |
| Av. | 60.89 | 113.21 | 351.96 | 848.73 | 1088.75 | 1386.38 |
| S.d. | (35.44) | (50.17) | (181.59) | (401.07) | (475.82) | (533.41) |
| # of cuts | | | | | | |
| Av. | 30.2 | 35.4 | 38.7 | 43.3 | 44.3 | 44.8 |
| S.d. | (8.02) | (6.30) | (10.61) | (8.58) | (8.80) | (5.78) |
| # of vertices | | | | | | |
| Av. | 788.0 | 959.1 | 1112.9 | 1387.4 | 1490.8 | 1476.7 |
| S.d. | (404.71) | (305.09) | (537.20) | (465.97) | (481.93) | (318.37) |

**Computational Experiments**

Here we will report the results of numerical experiments. We solved the following rank $p$ bilinear programming problem:

$$(3.23)\quad (BP)\left|\begin{array}{ll} \text{minimize} & \sum_{i=1}^{p} c_i^t x \cdot d_i^t y \\ \text{subject to} & A_1 x \geq b_1, \quad x \geq 0, \\ & A_2 y \geq b_2, \quad y \geq 0, \end{array}\right.$$

where $c_i, d_i, x, y \in R^n$, $A_1, A_2 \in R^{m \times n}$, $b_1, b_2 \in R^m$. All of these elements are randomly generated over the unit interval $[0, 1]$. The algorithm was coded in C language and was tested on a SUN4/75 workstation. We solved ten problems for each size. The tolerance was always fixed at $\epsilon = 10^{-5}$. The following tables show the average number of CPU time in seconds, those of cuts, those of vertices and their standard deviations, respectively. The number of cuts corresponds to the number of convex quadratic programs solved for each example. Also, the number of vertices corresponds to that of linear programming subproblems $F(\xi)$. Table 3.1,3.2 and 3.3 show the results when $p$ is 4, 5 and 6, respectively. Table 3.4 shows the results when $(m, n) = (80, 60)$ and $p$ ranges from 3 to 6.

We see from these tables that the number of cuts and vertices grows quite slowly as the size of $m$ or $n$ grows, while those of cuts and vertices mainly depend on $p$. In other words, computational time is almost dominated by that needed for solving convex QP (3.21) and subproblems $F(\xi)$. The efficiency of the algorithm would be improved by more elaborate implementation of algorithms for

Table 3.3: Results of (3.23) when $p = 6$ and $\epsilon = 10^{-5}$

| $m$ | 30 | 30 | 50 | 50 | 80 |
|---|---|---|---|---|---|
| $n$ | 20 | 40 | 40 | 60 | 60 |
| CPU time (in seconds) | | | | | |
| Av. | 113.59 | 463.81 | 834.02 | 997.70 | 2207.06 |
| S.d. | (90.06) | (534.87) | (672.50) | (613.65) | (1013.95) |
| # of cuts | | | | | |
| Av. | 31.6 | 41.3 | 46.4 | 47.1 | 54.9 |
| S.d. | ( 5.48) | ( 9.03) | (10.42) | ( 7.80) | ( 8.84) |
| # of vertices | | | | | |
| Av. | 2133.3 | 3688.0 | 4641.7 | 5077.0 | 6315.0 |
| S.d. | (898.32) | (1920.59) | (2035.60) | (1847.97) | (1953.47) |

Table 3.4: Results of (3.23) when $(m, n) = (80, 60)$ and $\epsilon = 10^{-5}$

| $p$ | 3 | 4 | 5 | 6 |
|---|---|---|---|---|
| CPU time (in seconds) | | | | |
| Av. | 75.72 | 144.71 | 351.96 | 2207.06 |
| S.d. | (26.56) | (30.43) | (181.59) | (1013.95) |
| # of cuts | | | | |
| Av. | 17.7 | 27.1 | 38.7 | 54.9 |
| S.d. | ( 3.44) | ( 3.11) | (10.61) | ( 8.84) |
| # of vertices | | | | |
| Av. | 62.6 | 255.4 | 1112.9 | 6315.0 |
| S.d. | (16.64) | (46.04) | (537.20) | (1953.47) |

solving (3.21).

Table 3.5 shows the results when $\epsilon$ ranges from $10^{-2}$ to $10^{-6}$ while $m, n, p$ are fixed. In this table, the error corresponds to the ratio of the $\epsilon$-optimal value to that obtained by the exact algorithm (Algorithm B) we will show in the following sections. We see from this table that we can obtain an almost exact solution when $\epsilon$ is $10^{-5}$.

Based upon these data, we conclude that Algorithm A is fairly efficient for the lower rank bilinear programming problems.

Table 3.5: Results of (3.23) when $p = 4$, $(m, n) = (60, 80)$

| $\epsilon$ | $10^{-2}$ | $10^{-3}$ | $10^{-4}$ | $10^{-5}$ | $10^{-6}$ |
|---|---|---|---|---|---|
| Error | | | | | |
| Av. | 1.04240 | 1.01997 | 1.00100 | 1.00013 | 1.00011 |
| S.d. | (0.00020) | (0.00111) | (0.05275) | (0.03275) | (0.00021) |
| # of cuts | | | | | |
| Av. | 22.6 | 26.1 | 27.1 | 29.7 | 32.5 |
| S.d. | (5.78) | (7.13) | (7.11) | (7.27) | (8.04) |

## 4. An Exact and Finitely Convergent Algorithm

Algorithm developed in the previous section generates a sequence of feasible solutions $(x^k, y^k)$which converges to an $\epsilon-$ optimal solution where $\epsilon > 0$. In this section, we propose an exact and finitely convergent outer approximation algorithm for solving the problem (2.7). Let $D = (D_B, D_N)$ be the partition of the matrix $D$, where $D_B \in R^{(p+1)\times(p+1)}$ is nonsingular, and $D_N \in R^{(p+1)\times(n_2-p-1)}$. Let us represent (2.10) in accordance with this partition:

(4.24) $\xi = D_B y_B + D_N y_N$,

where $y_B \in R^{p+1}$, $y_N \in R^{n_2-p-1}$. Also, let us represent $Y$ accordingly:

(4.25) $A_{2B} y_B + A_{2N} y_N \leq b_2$.

By substituting

(4.26) $y_B = {D_B}^{-1}(\xi - D_N y_N)$

into (4.25), we can rewrite $G$ as follows:

(4.27) $G = \{\xi \mid {}^{\exists} y_N \text{ such that } C\xi + E y_N - b_2 \leq 0\}$,

where $C = A_{2B} D_B^{-1}$, $E = A_{2N} - A_{2B} D_B^{-1} D_N$.

**Theorem 4.4** *$G$ can be expressed by finitely many inequalities as follows:*

(4.28) $G = \{\xi \mid {v^i}^t (C\xi - b_2) \leq 0,\ i = 1, 2, \dots r\}$,

*where $v^i$, $i = 1, 2, \dots r$ are the generators of the polyhedral cone*

(4.29) $\mathcal{C} = \{u \mid u^t E = 0,\ u \geq 0\}$.

**Proof** The lemma of Farkas and Minkowski [14] states:

$\xi$ belongs to $G$ if and only if

(4.30) $u^t(C\xi - b_2) \leq 0,\quad {}^{\forall} u \geq 0$ such that $u^t E = 0$.

Since $\mathcal{C}$ is a polyhedral cone, every $u \in \mathcal{C}$ can be expressed as a finite combination with non-negative coefficients of $v^i$, $i = 1, 2, \dots r$. □

Now we will introduce another cutting procedure. Let us assume that we have the $k$th relaxation as we defined in previous section.

(4.31) $(BP)^k \left| \begin{array}{ll} \text{minimize} & F(\xi) \\ \text{subject to} & \xi \in G^k, \end{array}\right.$

where $G^0 \supset G^k \supseteq G$.

Let $\xi^k$ be an extreme optimal solution of $(BP)^k$ and define the following linear programming problem:

(4.32) $\left| \begin{array}{ll} \text{maximize} & u^t(C\xi^k - b_2) \\ \text{subject to} & u^t E = 0,\ u \geq 0. \end{array}\right.$

**Theorem 4.5** *If $\xi^k$ belongs to $G$ then an optimal value of (4.32) is zero. Otherwise problem (4.32) has infinite optimal value and an associated direction $u^k$.*

**Proof** It is easy to see that problem (4.32) has either a zero optimal value or an infinite solution with an associated direction $v$. In former case $\xi^k$ satisfies all inequalities of (4.28). In latter case the constraint

(4.33) ${u^k}^t(C\xi - b_2) \leq 0$

is violated at $\xi^k$, which implies that $\xi^k \notin G$. □

Then we cut off $\xi^k$ by adding (4.33) to $G^k$.

We can construct an alternative outer approximation procedure for solving (2.16).

**Algorithm B**

**Step1** Let $k = 0$.

**Step2** Compute an optimal solution $\xi^k$ of the relaxation problem $(\mathrm{BP})^k$.

**Step3** If the optimal value of (4.32) is zero then stop. Otherwise, add the new constraint (4.33) to $G^k$. Let $k = k + 1$ and return to Step 2.

Table 4.1: Results of (3.23) when $p = 3$

| $m$ | | 50 | 50 | 60 | 80 | 100 | 100 |
|---|---|---|---|---|---|---|---|
| $n$ | | 40 | 60 | 80 | 100 | 100 | 120 |
| CPU time (in seconds) | | | | | | | |
| Algorithm A | Av. | 12.17 | 22.93 | 64.06 | 151.49 | 309.54 | 430.30 |
| | S.d. | (4.89) | (9.98) | (42.69) | (43.43) | (98.06) | (125.30) |
| Algorithm B | Av. | 14.16 | 34.92 | 70.52 | 191.55 | 432.53 | 531.18 |
| | S.d. | (5.60) | (19.81) | (31.52) | (68.19) | (251.49) | (208.48) |
| # of cuts | | | | | | | |
| Algorithm A | Av. | 14.4 | 15.6 | 18.1 | 17.9 | 20.7 | 20.0 |
| | S.d. | (2.24) | (3.53) | (5.79) | (2.84) | (4.47) | (4.49) |
| Algorithm B | Av. | 40.7 | 58.5 | 55.5 | 62.7 | 87.4 | 86.1 |
| | S.d. | (13.36) | (27.08) | (21.75) | (17.39) | (38.91) | (30.02) |
| # of vertices | | | | | | | |
| Algorithm A | Av. | 51.7 | 51.5 | 63.8 | 62.9 | 76.2 | 73.6 |
| | S.d. | (14.90) | (15.16) | (28.98) | (14.26) | (22.07) | (23.61) |
| Algorithm B | Av. | 187.5 | 274.0 | 255.4 | 299.7 | 442.4 | 429.0 |
| | S.d. | (68.58) | (137.82) | (121.46) | (89.89) | (217.31) | (154.16) |

□

**Theorem 4.6** *The algorithm* **B** *generates an exact optimal solution $\xi^*$ of (2.16) in a finite number of iterations.*

**Proof** The number of constraints of $G$ is finite and generated constraints are all different. □

**Computational Experiments**

We will briefly report the result of computational tests. We solved the same problems as in previous section. Tables 4.1 and 4.2 show some of the statistics when $m$ and $n$ are varied while keeping $p$ constant.

We see from these tables that the number of cuts of Algorithm B grows moderately as the size of the problems grows. The number of cuts generated by Algorithm B is about five times as many as that of Algorithm A. It turns out that Algorithm B is less efficient than Algorithm A. In Algorithm B, all inequalities which cut off $\xi^k \notin G$ could be generated. When we found several infinite directions associated with some basis of (4.32) we chose the farthest cut from $\xi^k$ among those. Not all infinite directions of (4.32) were considered. It appears that the efficiency depends upon the choice of them. How to find a good cut is yet to be analyzed.

## 5. Conclusion

In this paper, we proposed two outer approximation algorithms for rank $k$ bilinear programming problems. The results of computational experiments show that Algorithm A is more efficient than Algorithm B when the rank is less than five.

In Algorithm A, a cut is generated by solving a convex quadratic problem (3.21) which requires a lot of computational effort. However the total number of cuts generated by Algorithm A is smaller than that of Algorithm B. This means that a deep cut can be generated by convex quadratic problem (3.21). The same approach can be applied to other cutting plane procedures and may turn out to be effective. This point will be pursued further in the subsequent papers.

Infinite rays of problem (4.32) can be generated easily. In general, however, many infinite rays are generated, so that it is necessary to find a method to choose the most efficient one among

Table 4.2: Results of (3.23) when $p = 4$

| | | | | | | |
|---|---|---|---|---|---|---|
| $m$ | | 30 | 50 | 50 | 60 | 80 |
| $n$ | | 20 | 40 | 60 | 80 | 100 |
| CPU time (in seconds) | | | | | | |
| Algorithm A | Av. | 2.41 | 33.38 | 74.01 | 143.28 | 446.34 |
| | S.d. | ( 0.69) | (17.46) | (52.74) | (64.67) | (278.09) |
| Algorithm B | Av. | 4.45 | 118.00 | 164.70 | 424.38 | 1437.94 |
| | S.d. | ( 3.13) | (59.01) | (78.59) | (370.20) | (767.70) |
| # of cuts | | | | | | |
| Algorithm A | Av. | 16.6 | 24.5 | 28.5 | 29.2 | 33.1 |
| | S.d. | (2.73) | ( 5.73) | ( 6.52) | ( 6.63) | ( 9.54) |
| Algorithm B | Av. | 32.2 | 113.7 | 115.8 | 127.5 | 173.2 |
| | S.d. | (12.27) | (36.27) | (31.10) | (50.58) | (43.48) |
| # of vertices | | | | | | |
| Algorithm A | Av. | 115.7 | 218.6 | 270.9 | 289.0 | 350.0 |
| | S.d. | (38.19) | (91.93) | (91.55) | (99.45) | (161.77) |
| Algorithm B | Av. | 366.7 | 1888.0 | 2132.5 | 2250.3 | 3145.6 |
| | S.d. | (208.95) | (785.96) | (751.59) | (1047.95) | (1055.58) |

them. The way how to choose it is yet to be analyzed.

## References

[1] Gallo, G. and Ülkücü, A. (1977), "Bilinear Programming: An Exact Algorithm," *Mathematical Programming*, **12**, 173–194.

[2] Geoffrion, A. (1967), "Solving Bicriterion Mathematical Programs," *Oper. Res.*, **15**, 39–54.

[3] Henderson, J. M. and Quandt, R. E. (1971), *Microeconomics Theory*, McGraw-Hill, New York.

[4] Horst, R. and Tuy, H. (1990), *Global Optimization: Deterministic Approaches*, Springer-Verlag.

[5] Konno, H. (1971), "Bilinear Programming, Part 1 and 2," Technical Report No.71-9,71-10, Dept. of OR, Stanford University.

[6] Konno, H. (1976), "A Cutting Plane Algorithm for Solving Bilinear Programs," *Mathematical Programming*, **11**, 14–27.

[7] Konno, H. and Inori, M. (1988), "Bond Portfolio Optimization by Bilinear Fractional Programming," *J. Oper. Res. Soc. of Japan*, **32**, 143–158.

[8] Konno, H. and Kuno, T. (1990), "Generalized Linear Multiplicative and Fractional Programming," *Annals of Operations Research*, **25**, 147–162.

[9] Konno, H., Kuno, T. and Yajima, Y. (1992), "Parametric Simplex Algorithms for a Class of NP-Complete Problems Whose Average Number of Steps is Polynomial," *Computational Optimization and Applications*, **1**, 227–239.

[10] Konno, H., Pliska, S. R. and Suzuki, K. (1992), "Optimal Portfolios with Asymptotic Criteria, " IHSS Report 92-46, Institute of Human and Social Sciences, Tokyo Institute of Technology (to appear in *Annals of Operations Research*).

[11] Konno, H., Yajima, Y. and Matsui, T. (1991), "Parametric Simplex Algorithm for Solving a Special Class of Nonconvex Minimization Problems," *J. of Global Optimization*, **1**, 65–81.

[12] Majthay, A. and Whinston, A. B. (1974), "Quasi-concave Minimization Subject to Linear Constraints," *Discrete Math.*, **9**, 35–59.

[13] Maling, K., Mueller, S. H. and Heller, W. R. (1982), "On Finding Most Optimal Rectangular Package Plans," in *Proceedings of the 19th Design Automatic Conference*, 663–670.

[14] Minoux, M. (1986), *Mathematical Programming*, John Wiley, New York.

[15] Pardalos, P. M. and Vavasis, S. A. (1991), "Quadratic Programming with One Negative Eigenvalue is NP-hard," *J. of Global Optimization*, **1**, 15–22.

[16] Rosen, J. B. and Pardalos, P. M. (1986), "Global Minimization of Large-Scale Constrained Concave Quadratic Problems by Separable Programming," *Mathematical Programming*, **34**, 163–174.

[17] Shamir, R. (1987), "The Efficiency of the Simplex Method: A Survey," *Management Science*, **33**, 301–334.

[18] Tuy, H. (1964), "Concave Programming under Linear Constraints," *Soviet Mathematics*, **5**, 1437–1440.

[19] Vaish, H. and Shetty, C. M. (1976), "The Bilinear Programming Problem," *NRLQ*, **23**, 303–309.

[20] Yajima, Y. and Konno, H. (1991), "Efficient Algorithms for Solving Rank Two and Rank Three Bilinear Programming Problems," *J. of Global Optimization*, **1**, 155–171.

Yasutoshi YAJIMA
Department of Mathematical and Computing Sciences,
Tokyo Institute of Technology,
O-okayama, Meguro-ku, Tokyo 152 JAPAN