J Glob Optim (2007) 38:421–436
DOI 10.1007/s10898-006-9091-3

ORIGINAL ARTICLE

Accelerating convergence of cutting plane algorithms for disjoint bilinear programming

Xiaosong Ding · Faiz Al-Khayyal

Received: 04 August 2006 / Accepted: 28 August 2006 / Published online: 5 October 2006

Abstract This paper presents two linear cutting plane algorithms that refine existing methods for solving disjoint bilinear programs. The main idea is to avoid constructing (expensive) disjunctive facial cuts and to accelerate convergence through a tighter bounding scheme. These linear programming based cutting plane methods search the extreme points and cut off each one found until an exhaustive process concludes that the global minimizer is in hand. In this paper, a lower bounding step is proposed that serves to effectively fathom the remaining feasible region as not containing a global solution, thereby accelerating convergence. This is accomplished by minimizing the convex envelope of the bilinear objective over the feasible region remaining after introduction of cuts. Computational experiments demonstrate that augmenting existing methods by this simple linear programming step is surprisingly effective at identifying global solutions early by recognizing that the remaining region cannot contain an optimal solution. Numerical results for test problems from both the literature and an application area are reported.

Keywords Linear programming · Bilinear programming · Cutting plane · Polar cuts · Lower bounding

X. Ding
Department of Information Technology and Media, Mid-Sweden University, 85170 Sundsvall, Sweden
e-mail: dingxiaosong@bfsu.edu.cn

Present Address:
X. Ding (✉)
School of International Business, Beijing Foreign Studies University 100089, Beijing, P.R.China
e-mail: dingxiaosong@bfsu.edu.cn

F. Al-Khayyal
School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0205, USA
e-mail: faiz.alkhayyal@isye.gatech.edu

1 Introduction

An important class of hard non-convex programs that has many applications is the bilinear program (BLP) with disjoint constraints [5,8,14,16,17,22]. Mathematically, a disjoint BLP problem can be stated as

$$\begin{aligned} &\min f(x,y) = c^t x + d^t y + x^t C y, \\ &\text{s.t.} x \in X_0 = \{x \in R^{n_1} : A_1 x = b_1, x \geq 0\}, \\ &\quad y \in Y_0 = \{y \in R^{n_2} : A_2 y = b_2, y \geq 0\}, \end{aligned} \tag{1}$$

where X_0 and Y_0 are bounded polyhedral sets.

Several previous studies [1,5,11,13,27] have investigated the structural properties of (1), and many solution approaches have been proposed. Concavity cuts [24] were utilized in several early cutting plane algorithms [10,13]. The question of convergence was investigated by several authors (e.g., [11,31]), and this was finally settled in the affirmative [18]. Nevertheless, it has been shown that concavity cuts are uniformly dominated by polar cuts which have been employed in other cutting plane methods [7,21,28]. In general, cutting plane methods converge slowly near an optimal solution because successive cuts become nearly parallel thereby eliminating only a very small part of the feasible region in each iteration [11,26].

Another solution strategy for bilinear programming is branch and bound. Many such methods have been developed for solving both disjoint and jointly constrained BLP [1,2,6,9,20]. In general, branch and bound procedures take a long time to verify that an incumbent solution is actually the global optimum. Other less common approaches include methods based on an annexation strategy, linear complementarity problems and linear max–min reformulations [12,23,27,29,30]. More recently, [4] proposed a method that combines concavity cuts with the branch and bound procedure developed in [6]. This method first uses concavity cuts to reduce the feasible region of (1), and then it carries out the branch and bound method over the reduced feasible region. Unlike the traditional cutting plane methods, this method adds concavity cuts to both X_0 and Y_0, thereby making the computational load heavier than would be the case if cuts were generated for only one polyhedron, say, X_0. Comprehensive studies about BLP can be found in [11,25,26].

In this paper, we develop two procedures that combine the generation of polar cuts with the computation of lower bounds, using the technique proposed in [1,2], to achieve fast convergence. The next section describes how the existing cutting plane methods are both modified and augmented to accelerate convergence. In particular, our refinement avoids the (computationally expensive) construction of disjunctive face cuts and only generates polar cuts. Combining this change with our lower bounding techniques preserves convergence and leads to measurable speedups in convergence. This is illustrated in Sect. 3 by numerical experiments which clearly show the advantages of incorporating the bounding step into the procedure.

2 Optimization

To solve the disjoint BLP program, global optimization strategies will be used. A general framework is to iterate between a global (bounding) phase of systematically exploring the feasible region, subset by subset and a local (improvement) phase designed to determine a local optimizer starting from an approximate solution [11, 26]. As will be demonstrated later, the two global optimization algorithms described herein capture the spirit of this framework and find either an exact global minimizer or an epsilon-global minimizer, with a pre-specified epsilon-tolerance on the optimal objective value. Moreover, it is also possible to stop at any feasible point with a known worst case error bound on how far the incumbent solution is away from global optimality as measured by objective value difference.

The most important property of a disjoint BLP is that, even though $f(x,y)$ may not be quasi-concave, there exists an extreme point $\bar{x} \in X_0$ and an extreme point $\bar{y} \in Y_0$ such that $(\bar{x},\bar{y})$ is an optimal solution of problem (1) (see, e.g., [1,11,13]).

2.1 Local optimization

The solution property and the structure of a disjoint BLP program itself suggest a linear programming (LP) based vertex following algorithm that converges to a *Karush–Kuhn–Tucker* point [13].

Definition 1 Consider $P : \min f(x)$ subject to $x \in S$, where S is a compact polyhedral set and f is non-convex. A *local star minimizer* (LSM) of P is defined as a point $\bar{x}$ such that $f(\bar{x}) \le f(x)$ for each $x \in N_S(\bar{x})$, where $N_S(\bar{x})$ denotes the set of extreme points in S that are adjacent to $\bar{x}$.

For a disjoint BLP, an extreme point is adjacent to $(\bar{x},\bar{y})$ if and only if it is of the form either $(x^i,\bar{y})$ or $(\bar{x},y^i)$, where $x^i \in N_{X_0}(\bar{x})$ and $y^i \in N_{Y_0}(\bar{y})$.

Definition 2 An extreme point $(\bar{x},\bar{y})$ is called a *pseudo-global minimizer* (PGM) if $f(\bar{x},\bar{y}) \le f(x,y)$ for each $x \in B_\delta(\bar{x}) \cap X_0$ and for each $y \in Y_0$, where $B_\delta(\bar{x})$ is a δ neighbourhood around $\bar{x}$.

An LP based procedure to obtain a PGM is the following [13].

Local Optimization Algorithm 1 (LOA_1):

(1) Find a feasible extreme point $\tilde{x}^1$ in X_0^i, where X_0^i represents the reduced feasible region in ith iteration after all cuts have been added.
(2) [a] Solve: $\min\{f(\tilde{x}^1,y)|y \in Y_0\}$, to yield an optimal $\tilde{y}^1$;
[b] Solve: $\min\{f(x,\tilde{y}^1)|x \in X_0\}$, to yield an optimal $\tilde{x}^2$;
Set $\tilde{x}^1 \leftarrow \tilde{x}^2$ and repeat step (2) until it converges to an LSM $(\bar{x},\bar{y})$.
(3) Suppose $\bar{x}$ is non-degenerate and let $\hat{x} \in N(\bar{x})$ be such that
$$f(\hat{x},\hat{y}) = \min_{y \in Y_0} f(\hat{x},y) < \min_{y \in Y_0} f(\bar{x},y) = f(\bar{x},\bar{y}).$$
If no such point exists, terminate with $(\bar{x},\bar{y})$ as a PGM.
(4) Go to step (2[b]) with $\tilde{y}^1 \leftarrow \hat{y}$.

The LOA_1 can be easily implemented, but it does not discriminate between the extreme points belonging to the original feasible region X_0, and those induced by the added cuts. But to solve (1), we should endeavor to reach an extreme point of X_0, which appears rather difficult [11].

However, we appeal to the efficient face identification routine (EFIR) [15] for this purpose. The key idea is to identify the extreme faces of X_0 relative to the cuts. Suppose s cuts, $Dx \leq d$, have been added to X_0 and let the set of feasible points be $Q = \{x \in R^{n_1} : Dx + Ix_s = d, x_s \geq 0\}$, where x_s denotes the vector of slack variables $\{x_{n_1+1}, \ldots, x_{n_1+s}\}^t$ and I denotes an identity matrix.

Definition 3 Let X_0 be a convex subset in R^{n_1}. A non-empty subset F of X_0 is called a (proper) face of X_0 if there exists a supporting hyperplane H of X_0 such that $F = X_0 \cap H$.

Now let $N = \{1, \ldots, n_1\}$ denote the index set of the original set of variables (key variables), and let $S = \{n_1+1, \ldots, n_1+s\}$ denote the index set of the slack variables of the s cuts (non-key variables). For a subset $Z \subset N$, let $F_Z = \{x \in X_0 : x_j = 0 \text{ for } j \in Z\}$.

Definition 4 Let F_Z be a face of X_0 such that $F_Z \cap Q \neq \emptyset$. Then F_Z is an extreme face of X_0 relative to Q if for each $k \in N$, $x \in F_{Z \cup k} \neq F_Z$ implies $x \notin Q$.

Given a set $Z_0 \subset N$, an extreme face of X_0 can be identified by sequentially adding indices to the set Z_0 subject to a revision of the basis entry rule in the simplex method as "only a non-key variable x_j, $j \in S$, is eligible to enter the basis." It has been proved that this procedure either finds an extreme face or indicates that no such face exists [15].

Definition 5 Let Q be the region feasible to the s cuts generated so far and let $(\bar{x}, \bar{y})$ be an extreme point of $X_0 \times Y_0$ such that $\bar{x} \in Q$ and $\min_{y \in Y_0} f(\bar{x}, y) = f(\bar{x}, \bar{y})$. Consider a basis B of (1) representing $\bar{x}$. Then $(\bar{x}, \bar{y})$ is said to be a *weak pseudo-global minimum* (WPGM) relative to the basis B if for each $\hat{x} \in N(\bar{x})$ such that $\hat{x} \in Q$, we have $\min_{y \in Y_0} f(\hat{x}, y) \geq f(\bar{x}, \bar{y})$.

Given a simplex tableau representing a non-degenerate extreme point x^e of X_0. If $x^e \in Q$, the set $N(x^e) \cap Q$ can be readily obtained from the current tableau as points resulting from single pivots which involve the exchange of a key variable for another key variable.

Local Optimization Algorithm 2 (LOA_2) Let $k = 0$.

(1) Let $\hat{x} \in N(x^k) \cap Q$ be such that

$$\min_{y \in Y_0} f(\hat{x}, y) < \min_{y \in Y_0} f(x^k, y) = f(x^k, y^k).$$

If no such point exists, terminate with $(\bar{x}, \bar{y}) = (x^k, y^k)$ as a WPGM.

(2) Increase k by 1 and go to step (1) with $x^{k+1} = \hat{x}$.

The difference between LOA_1 and LOA_2 lies in that LOA_2 tries to restrict the search to the extreme points in X_0 that are feasible to the added cuts rather than to the whole set of extreme points in X_0^i.

2.2 Global optimization

2.2.1 Cutting plane methods

Given a PGM or WPGM located by LOA_1 or LOA_2, respectively, we employ polar cuts to cut off local vertex solutions.

Assume $\bar{x}$ is a non-degenerate extreme point of X_0; let $p = n_1 - m$, where m is the number of rows in A_1 in (1); and let $x_j, j \in \overline{N}$, be the p non-basic variables at $\bar{x}$, where $\overline{N}$ is the index set for the non-basic variables. Then X_0 has precisely p distinct edges incident to $\bar{x}$. Each half line $\xi^j = \{x : x = \bar{x} - \bar{a}^j \lambda_j, \lambda_j \geq 0\}, j \in \overline{N}$, contains exactly one such edge [7].

Definition 6 The *generalized reverse polar* of Y_0 for a given scalar α is given by $Y_0(\alpha) = \{x : f(x,y) \geq \alpha\}$ for all $y \in Y_0$.

Let $(\bar{x}, \bar{y})$ be a PGM or WPGM, let the rays ξ^j be defined as above, let α be the current best objective value (CBOV) of $f(x,y)$, and let $\overline{\lambda}_j$ be defined by

$$\overline{\lambda}_j = \begin{cases} \max\{\lambda_j : f(\bar{x} - a^j \lambda_j, y) \geq \alpha \text{ for all } y \in Y_0\} & \text{if } \xi^j \not\subset Y_0(\alpha), \\ -\max\{\lambda_j : f(\bar{x} + a^j \lambda_j, y) \geq \alpha \text{ for some } y \in Y_0\} & \text{if } \xi^j \subset Y_0(\alpha). \end{cases}$$

Then the inequality $\sum_{j \in \overline{N}} x_j / \overline{\lambda}_j \geq 1$ determines a valid cutting plane [21,28]. Each $\overline{\lambda}_j$ can be computed by an efficient modification of Newton's method [21].

We are now ready to present two established pure cutting plane algorithms which adopt LOA_1 and LOA_2, respectively. There are three common stopping criteria.

Terminating Rules for Pure Cutting Plane Methods (TRP):

(a) There exists no $\overline{\lambda}_j$ such that $\xi^j \not\subset Y_0(\alpha)$;
(b) There exists $\overline{\lambda}_j$ such that $\xi^j \not\subset Y_0(\alpha)$, but there also exists $\overline{\lambda}_j = 0$ such that $\xi^j \subset Y_0(\alpha)$;
(c) $X_0^i = \emptyset$.

The TRP (a) and TRP (b) are stopping criteria induced by polar cuts, and TRP (c) is the stopping criterion for any cutting plane algorithm. In the following algorithms, obj_i represents the CBOV in ith iteration.

Algorithm 1 (Alg_1)

(1) Let $\text{obj}_0 = +\infty$ and $\{(\hat{x}^0, \hat{y}^0)\} = \emptyset$; let an epsilon tolerance, ε, be a prescribed small positive number; set $i = 1$ and $X_0^i = X_0$.
(2) If TRP (c) is satisfied, terminate with obj_{i-1} as the global minimum and $(\hat{x}^{i-1}, \hat{y}^{i-1})$ as the corresponding global minimizer.
(3) Find a PGM $(\bar{x}^i, \bar{y}^i)$ by using LOA_1 with $X_0 \leftarrow X_0^i$; change obj_i and $(\hat{x}^i, \hat{y}^i)$ by setting $\text{obj}_i = \min\{\text{obj}_{i-1}, f(\bar{x}^i, \bar{y}^i)\}$ and $(\hat{x}^i, \hat{y}^i) = \text{argmin}\{\text{obj}_{i-1}, f(\bar{x}^i, \bar{y}^i)\}$, respectively.

(4) Use the modification of Newton's procedure [21] to obtain $\overline{\lambda}_j, j \in \overline{N}$; generate an appropriate polar cut; define $X_0^{i+1} = X_0^i \cap H^+(\overline{x}^i)$, where $H^+(\overline{x}^i)$ is the feasible half space defined by cutting off $\overline{x}^i$
(5) If either TRP (a) or TRP (b) is satisfied, terminate with obj_i as the global minimum and $(\hat{x}^i, \hat{y}^i)$ as the corresponding global minimizer.
(6) Set $i \leftarrow i+1$ and return to (2).

In Alg_1, we separate steps (2) and (5) for the three terminating conditions even though they could be checked within one step. The reason is that we do not know whether TRP (a) or (b) is true before we generate the first polar cut.

Algorithm 2 (Alg_2)

(1) Find a PGM $(\overline{x}^0, \overline{y}^0)$ by using LOA_1; set $\text{obj}_0 = f(\overline{x}^0, \overline{y}^0)$; set $(\hat{x}^0, \hat{y}^0) = (\overline{x}^0, \overline{y}^0)$; let an epsilon tolerance, ε, be a prescribed small positive number; set $i = 1$ and $X_0^i = X_0$.
(2) Use the modification of Newton's procedure [21] to obtain $\overline{\lambda}_j, j \in \overline{N}$; generate an appropriate polar cut; define $X_0^{i+1} = X_0^i \cap H^+(\overline{x}^i)$.
(3) If either TRP (a) or (b) is satisfied, terminate with obj_{i-1} as the global minimum and $(\hat{x}^{i-1}, \hat{y}^{i-1})$ as the corresponding global minimizer.
(4) Apply EFIR [15] in an attempt to find a vertex of X_0 that satisfies all cuts generated thus far.
   I. If TRP (c) is satisfied, terminate with obj_{i-1} as the global minimum and $(\hat{x}^{i-1}, \hat{y}^{i-1})$ as the corresponding global minimizer.
   II. If the point found by EFIR is an extreme point of X_0 feasible to the cuts, locate a WPGM by using LOA_2.
   III. If the point found by EFIR is not an extreme point of X_0 feasible to the cuts, locate a PGM by using LOA_1 with $X_0 \leftarrow X_0^i$.

   For Cases II and III, set $\text{obj}_i = \min\{\text{obj}_{i-1}, f(\overline{x}^i, \overline{y}^i)\}$ and $(\hat{x}^i, \hat{y}^i) = \text{argmin}\{\text{obj}_{i-1}, f(\overline{x}^i, \overline{y}^i)\}$, respectively.
(5) Set $i \leftarrow i+1$ and return to (2).

In Alg_2, we use LOA_1 to locate a PGM in step (1) because every PGM is a WPGM at this stage. When EFIR fails to indicate that the current point is a proper extreme point in X_0 feasible to the added cuts (i.e., not all non-basic variables are key variables), we do not turn to the generation of disjunctive cuts as in [21] due to the heavy computational burden. Intuitively, in a cutting plane procedure, EFIR should be effective in early stages and gradually appear inefficient because of the increasing number of extreme points induced by the cuts and the decreasing number of extreme points in X_0 removed by the cuts. Therefore, in order to generate only inexpensive cuts, our approach makes use of EFIR and LOA_2 whenever possible. When EFIR fails to locate a vertex of X_0, we fall back to LOA_1 and continue to generate a polar cuts.

A global solution found by Alg_1 and Alg_2 cannot be confirmed until all remaining inferior extreme points of X_0 are cut off. Consequently, the basic idea is to repeatedly calculate a tight, but inexpensive, lower bound on the global optimum. Then at least we can tell how close the CBOV is to global optimality at any time during an exhaustive search process.

2.2.2 *Arithmetic intervals*

Consider x^tBy over the compact hyper-rectangle $\Omega = \{(x,y) : l \le x \le L, m \le y \le M\}$. Define $\Omega_{ij} = \{(x_i,y_j) : l_i \le x_i \le L_i, m_j \le y_j \le M_j\}$. In [1, 2], the convex and concave envelopes of x_iy_j over Ω_{ij} have been shown to be

$$\begin{aligned}\text{Vex}_{\Omega_{ij}}[x_iy_j] &= \max\{m_jx_i + l_iy_j - l_im_j, M_jx_i + L_iy_j - L_iM_j\},\\ \text{Cav}_{\Omega_{ij}}[x_iy_j] &= \min\{M_jx_i + l_iy_j - l_iM_j, m_jx_i + L_iy_j - L_im_j\}.\end{aligned} \tag{2}$$

Given a bounded disjoint BLP problem, for an entry with $b_{ij} > 0$ in the bilinear term x^tBy, we compute its convex envelope as $\text{Vex}[b_{ij}x_iy_j] = b_{ij}\text{Vex}[x_iy_j]$. For an entry with $b_{ij} < 0$, we compute the concave envelope as $\text{Cav}[|b_{ij}|x_iy_j] = |b_{ij}|\text{Cav}[x_iy_j]$. Then we can say

$$\begin{aligned} f(x,y) &= c^tx + d^ty + x^tCy\\ &= c^tx + d^ty + \sum b_{ij}x_iy_j\\ &= c^tx + d^ty + \sum_{b_{ij}>0} b_{ij}x_iy_j - \sum_{b_{ij}<0} |b_{ij}|x_iy_j\\ &\ge \sum_{b_{ij}>0} \text{Vex}[b_{ij}x_iy_j] - \sum_{b_{ij}<0} \text{Cav}[|b_{ij}|x_iy_j]. \end{aligned} \tag{3}$$

We use (3) to underestimate the optimal value of (1) over subsets of the feasible region that are in Ω. An important computational observation is that the tighter the lower and upper bounds imposed over x_i and y_j, the higher the underestimation generated by (3) over the partition set. Observe that minimizing $\sum_{(i,j)} \text{Vex}_{\Omega_{ij}}[x_iy_j]$ is equivalent to minimizing $\sum_{(i,j)} t_{ij}$ subject to the two additional constraints for each (i,j)

$$\begin{aligned} t_{ij} &\ge m_jx_i + l_iy_j - l_im_j,\\ t_{ij} &\ge M_jx_i + L_iy_j - L_iM_j \end{aligned}$$

and minimizing $\sum_{(i,j)}\{-\text{Cav}_{\Omega_{ij}}[x_iy_j]\}$ is equivalent to minimizing $\sum_{(i,j)} t_{ij}$ subject to

$$\begin{aligned} t_{ij} &\ge l_iM_j - M_jx_i - l_iy_j,\\ t_{ij} &\ge L_im_j - m_jx_i - L_iy_j. \end{aligned}$$

2.3 Mixed strategies

Six stopping rules have to be specified for the two improved cutting plane algorithms, in which bound_i represents the lower underestimation over X_0^i in the ith iteration.

Terminating Rules for Improved Cutting Plane Methods (TRI):

(a) $\text{bound}_i > \text{obj}_{i-1}$;
(b) $|\text{bound}_i - \text{obj}_{i-1}| \le \varepsilon$;
(c) $|\text{bound}_i - \text{obj}_i| \le \varepsilon$;
(d) $X_0^i = \emptyset$;
(e) There exists no $\overline{\lambda}_j$ such that $\xi^j \not\subset Y_0(\alpha)$;
(f) There exists $\overline{\lambda}_j$ such that $\xi^j \not\subset Y_0(\alpha)$, but there also exists $\overline{\lambda}_j = 0$ such that $\xi^j \subset Y_0(\alpha)$.

The TRIs (a), (b) and (c) are stopping rules induced by the lower bounding technique, and the other three (d) through (f) are the same as those in TRP. Terminating with TRI (b) or TRI (c) yields an epsilon-global minimum, while terminating with the other rules finds the exact global minimum. Incorporating the lower bounding technique and the stopping rules into Alg_1 and Alg_2, we obtain two global optimization algorithms Alg_3 and Alg_4 which improve on Alg_1 and Alg_2, respectively. These are summarized below.

Algorithm 3 (Alg_3)

(1) Let $\text{obj}_0 = +\infty$ and $\{(\hat{x}^0, \hat{y}^0)\} = \emptyset$; let an epsilon tolerance, ε, be a prescribed small positive number; set $i = 1$ and $X_0^i = X_0$.
(2) Update (or calculate directly for $i = 1$) the lower and upper bounds for each variable; compute the underestimation, bound_i, for X_0^i.
(3) If either TRIs (a), or (b), or (d) is satisfied, terminate with obj_{i-1} as the global minimum and $(\hat{x}^{i-1}, \hat{y}^{i-1})$ as the corresponding global minimizer.
(4) Find a PGM $(\overline{x}^i, \overline{y}^i)$ by using LOA_1 with $X_0 \leftarrow X_0^i$; change obj_i and $(\hat{x}^i, \hat{y}^i)$ by setting $\text{obj}_i = \min\{\text{obj}_{i-1}, f(\overline{x}^i, \overline{y}^i)\}$ and $(\hat{x}^i, \hat{y}^i) = \text{argmin}\{\text{obj}_{i-1}, f(\overline{x}^i, \overline{y}^i)\}$, respectively.
(5) If TRI (c) is satisfied, terminate with obj_i as the global minimum and $(\hat{x}^i, \hat{y}^i)$ as the corresponding global minimizer.
(6) Use the modification of Newton's procedure [21] to obtain $\overline{\lambda}_j, j \in \overline{N}$; generate an appropriate polar cut; define $X_0^{i+1} = X_0^i \cap H^+(\overline{x}^i)$.
(7) If either TRI (e) or (f) is satisfied, terminate with obj_i as the global minimum and $(\hat{x}^i, \hat{y}^i)$ as the corresponding global minimizer.
(8) Set $i \leftarrow i + 1$ and return to (2).

Algorithm 4 (Alg_4)

(1) Find a PGM $(\overline{x}^0, \overline{y}^0)$ by using LOA_1; set $\text{obj}_0 = f(\overline{x}^0, \overline{y}^0)$; set $(\hat{x}^0, \hat{y}^0) = (\overline{x}^0, \overline{y}^0)$; let an epsilon tolerance, ε, be a prescribed small positive number; set $i = 1$ and $X_0^i = X_0$.
(2) Update (or calculate directly for $i = 1$) the lower and upper bounds for each variable; compute the underestimation, bound_i, for X_0^i.
(3) If TRIs (a) or (b) is satisfied, terminate with obj_{i-1} as the global minimum and $(\hat{x}^{i-1}, \hat{y}^{i-1})$ as the corresponding global minimizer.
(4) Use the modification of Newton's procedure [21] to obtain $\overline{\lambda}_j, j \in \overline{N}$; generate an appropriate polar cut; define $X_0^{i+1} = X_0^i \cap H^+(\overline{x}^i)$.
(5) If either TRI (e) or TRI (f) is satisfied, terminate with obj_{i-1} as the global minimum and $(\hat{x}^{i-1}, \hat{y}^{i-1})$ as the corresponding global minimizer.
(6) Try to find a starting point by using EFIR [15].
    I. If TRI (d) is satisfied, terminate with obj_{i-1} as the global minimum and $(\hat{x}^{i-1}, \hat{y}^{i-1})$ as the corresponding global minimizer.
    II. If the point is actually an extreme point in X_0 feasible to the cuts, locate a WPGM by using LOA_2.
    III. If the point is not an extreme point in X_0 feasible to the cuts, locate a PGM by using LOA_1 with $X_0 \leftarrow X_0^i$.

    For Cases II and III, set $\text{obj}_i = \min\{\text{obj}_{i-1}, f(\overline{x}^i, \overline{y}^i)\}$ and $(\hat{x}^i, \hat{y}^i) = \text{argmin}\{\text{obj}_{i-1}, f(\overline{x}^i, \overline{y}^i)\}$, respectively.

(7) If TRI (c) is satisfied, terminate with obj_i as the global minimum and $(\hat{x}^i, \hat{y}^i)$ as the corresponding global minimizer.
(8) Set $i \leftarrow i+1$ and return to (2).

The convergence proof for Alg_3 is provided below. With some minor modifications, the convergence proof for Alg_4 can be readily obtained.

Convergence proof (Alg_3) First, note that LOA_1 is finite so step (4) in Alg_3 yields exact solutions. Consider the sequence of PGMs $\{(\bar{x}^i, \bar{y}^i)\}$ generated and let $H(\bar{x}^i)$ be the cutting plane that eliminates $\bar{x}^i$. In step (7) of iteration i, the algorithm is terminated as the consequence of introducing polar cuts. In step (6) of iteration i, the algorithm is terminated if $X_0^i \cap H^+(\bar{x}^i) = \emptyset$ (actually detected in step (3) of iteration $i+1$). Otherwise, the cut $H(\bar{x}^i)$ is applied and a new PGM $(\bar{x}^{i+1}, \bar{y}^{i+1})$ is found where $\bar{x}^{i+1} \in X_0^i \cap H^+(\bar{x}^i)$ and $\bar{x}^i \notin H^+(\bar{x}^i)$. For $\varepsilon > 0$, it is possible for the process not to terminate by any of the six rules TRI(a) through TRI(f). An infinite sequence would then be generated, and we need to show that the sequence $\{\bar{x}^i\}$ has a limit point x^* such that $\lim_{i\to\infty} X_0^i \cap H^+(\bar{x}^i) = \emptyset$.

Since X_0 is a compact set, there exists a limit point x^* such that for a given $\epsilon \geq 0$ and a positive integer ν, $\|\bar{x}^i - x^*\| \leq \epsilon$ for infinitely many $i \geq \nu$. If $X_0^i \cap H^+(\bar{x}^i) \neq \emptyset$ for all $i \geq \nu$, then all subsequent PGMs $(\bar{x}^l, \bar{y}^l)$ generated will satisfy the condition $\bar{x}^l \in H^+(x^\nu)$ for all $l \geq \nu + 1$. From the definition of a PGM, $x^* \in B_\delta(x^*) \cap X_0$ and $\bar{x}^l \notin B_\delta(x^*)$ for some $\delta > 0$. Hence, $\|x^l - x^*\| \geq \delta$ for all $l \geq \nu+1$. This contradicts the statement that x^* is a limit point. Therefore, $\lim_{i\to\infty} X_0^i \cap H^+(\bar{x}^i) = \emptyset$ and the cutting plane algorithm is terminated.

Finite convergence The above convergence proof is essentially that for the pure cutting plane method Alg_1 if Alg_3 does not terminate finitely; (see [21,28]). In Alg_3, the introduction of the comparison between obj_i or obj_{i-1} and the underestimation of the optimal objective value simply accelerate termination and do not affect overall convergence in the limit. The two ways the algorithm terminates in a finite number of steps to an epsilon-optimal solution are described below. Moreover, when the global solution is unique, the incorporation of an additional step will guarantee convergence to an exact optimal solution in a finite number of steps.

I In step (3), if $\text{bound}_i > \text{obj}_{i-1}$ is satisfied, then, in the reduced feasible region, we cannot find a PGM with objective value better than CBOV. Hence, Alg_3 terminates after finitely many iterations with an exact global minimizer $(\hat{x}^{i-1}, \hat{y}^{i-1})$.

II In step (3), respectively, step (5), if either $|\text{bound}_i - \text{obj}_{i-1}| \leq \epsilon$, respectively, $|\text{bound}_i - \text{obj}_i| \leq \epsilon$, is satisfied, then the absolute difference between the objective value of the best feasible solution we have found and a lower bound on the global minimum is within a prescribed tolerance. Then Alg_3 terminates in finitely many iterations with an epsilon-global minimizer $(\hat{x}^{i-1}, \hat{y}^{i-1})$ in step (3), respectively $(\hat{x}^i, \hat{y}^i)$ in step (5).

III Finite convergence to a *unique* global minimizer can be guaranteed by introducing an additional step which requires the periodic solution of a linear program. This can be achieved by appealing to the result in [3,19]: if $\{z^k\} \subset \mathcal{S}$ satisfies $z^k \to z^*$ and $-\nabla\varphi(z^*) \in \text{int}\mathcal{N}(z^*)$, where $\text{int}\mathcal{N}(z^*)$ is the interior of the normal cone of convex set $\mathcal{S}$ at z^*, then there is a positive integer K such that for all $k > K$, the limit point x^* solves the convex program $\min\{\nabla\varphi(z^k)^t z : z \in \mathcal{S}\}$. If we set $\epsilon = 0$ and *Alg*$_3$ does not terminate in step (3), then let us assume that it converges to a unique limit point; i.e., we have $(\hat{x}^i, \hat{y}^i) \to (x^*, y^*)$. Moreover, (x^*, y^*) is an extreme point of $X_0 \times Y_0$ so that $\text{int}\mathcal{N}(x^*, y^*) \neq \emptyset$, where $\mathcal{N}(x^*, y^*)$

is the normal cone of $X_0 \times Y_0$ at (x^*, y^*), and $-\nabla f(x^*, y^*) \in int\mathcal{N}(x^*, y^*)$ by virtue of uniqueness of (x^*, y^*). It follows that (x^*, y^*) solves the linear program

$$\begin{aligned} \min f(x,y) &= (c + C\hat{y}^i)^t x + (d + C\hat{x}^i)^t y, \\ \text{s.t.}\ \ x \in X_0^i &= X_0^{i-1} \bigcap H^+(\overline{x}^{i-1}), \\ y \in Y_0 & \end{aligned} \tag{4}$$

for all $i > K$ for some K. Hence, modifying step (5) of Alg_3 to include solving linear program (4) will generate a sequence of solutions $\{(\tilde{x}^i, \tilde{y}^i)\}$. If bound$_i \geq f(\tilde{x}^i, \tilde{y}^i)$ then terminate with global minimizer $(\tilde{x}^i, \tilde{y}^i)$. If $f(\tilde{x}^i, \tilde{y}^i) \leq$ obj$_i$, then we may set our current best feasible solution to $(\hat{x}^i, \hat{y}^i) \leftarrow (\tilde{x}^i, \tilde{y}^i)$ and set obj$_i \leftarrow f(\tilde{x}^i, \tilde{y}^i)$. This extra linear program can be solved periodically; say, every τ iterations. For iterations $i = \tau, 2\tau, 3\tau, \ldots,$ set $\overline{x}^i \leftarrow \tilde{x}^i$ and proceed to step (6) to cut $\tilde{x}^i$ from X_0^i. For some finite ν, sufficiently large, we must have $x^\nu = x^*$ which would be cut off, so that obj$_i$ = obj$*$ for all $i > \nu$. Therefore, since bound$_{i+1} \geq$ bound$_i$ by virtue of $X_0^{i+1} \subset X_0^i$, it follows that for sufficiently large finite i we must have bound$_i >$ obj$*$ and Alg_3 terminates. □

In the worst case, the number of iterations for Alg_3 or Alg_4 will be equivalent to that of the corresponding pure cutting plane algorithm. Nonetheless, according to our numerical experiments, we observe this kind of situation seldom happens because actually the lower bounding technique always takes effect; i.e. one of the rules TRI(a) through TRI(c) stops the solution process.

3 Numerical example

Example 1 (Ex1)

$$\min \begin{bmatrix} 1 \\ -1 \end{bmatrix}^t \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -1 \\ 0 \end{bmatrix}^t \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} -2 & 1 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix},$$

$$s.t. \quad \begin{bmatrix} 1 & 4 \\ 4 & 1 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 8 \\ 12 \\ 12 \end{bmatrix}, \begin{bmatrix} 2 & 1 \\ 1 & 2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \leq \begin{bmatrix} 8 \\ 8 \\ 5 \end{bmatrix},$$

$$x_1, x_2, y_1, y_2 \geq 0.$$

In Ex1, there are four PGMs located by Alg_1 and three WPGMs located by Alg_2, respectively. Accordingly, Alg_1 cannot terminate until step (2) in the fifth iteration, while Alg_2 stops running at step (4.I) in the third iteration. TRP (c) is satisfied for both of them. Nevertheless, by incorporating the lower bounding technique, both algorithms can be terminated before X_0 is exhausted; i.e., before all extreme points are cut off. Detailed computational results are shown in Table 1, in which Iter represents the iteration index.

As noted in Table 1, the lower bounding technique generates a tight underestimation achieving the global minimum −25.0000. Both Alg_3 and Alg_4 require two fewer iterations than Alg_1 and Alg_2, respectively, with Alg_3 stopping in the second iteration of step (5) Alg_4 stopping in the first iteration of step (7).

Table 1 Results for example 1

Iter	Alg_1	Alg_3		Alg_2	Alg_4	
	obj	obj	bound	obj	obj	bound
0	–	–	–	−18.0000	−18.0000	–
1	−18.0000	−18.0000	−25.0000	−25.0000	−25.0000	−25.0000
2	−25.0000	−25.0000	−25.0000	−4.0000		
3	−18.1707					
4	−12.1935					

Example 2 (Ex2)
In (1), $c = d = 0$, $b_1 = b_2 = [10, 10, 10, 10]^t$ and

$$C = \begin{bmatrix} -3 & 1 & 0 & 1 \\ 1 & -4 & 2 & 0 \\ 0 & 2 & -4 & 1 \\ 1 & 0 & 1 & -3 \end{bmatrix}, \quad A_1 = A_2 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \end{bmatrix},$$

$$x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4 \geq 0.$$

The Ex2 has two global minimizers with the objective value −25.0000. Unlike Ex1, this time the lower bounding technique generates a relatively loose underestimation. Alg_1 stops at step (2) in the eighth iteration, while Alg_2 stops at step (4.I) in the sixth iteration. The TRP (c) is satisfied for both of them. Detailed computational results are shown in Table 2.

The underestimation starts with the value -35.0000, which is relatively far from the global minimum as compared with Ex1. It is gradually improved in the following several iterations. Finally, Alg_3 stops at step (3) in the fourth iteration with TRI (a) satisfied, thereby saving four iterations over Alg_1. The Alg_4 stops at step (3) in the fourth iteration with TRI (a) satisfied, which saves iterations over Alg_2. As observed in Table 2, the two global minimizers are located at a very early stage, and a pure cutting plane method like Alg_1 or Alg_2 will continue to perform the cutting procedure until X_0 is exhausted. However, these algorithms can be terminated earlier by embedding our proposed lower bounding technique.

Table 2 Results for example 2

Iter	Alg_1	Alg_3		Alg_2	Alg_4	
	obj	obj	bound	obj	obj	bound
0	–	–	–	−25.0000	−25.0000	–
1	−25.0000	−25.0000	−35.0000	−25.0000	−25.0000	−35.0000
2	−25.0000	−25.0000	−31.7301	−18.7500	−18.7500	−31.7301
3	−18.7500	−18.7500	−27.4695	−18.7500	−18.7500	−27.4695
4	−18.7500	−18.7500	−24.6436	0	0	−24.6436
5	−15.2439			−5.5302		
6	−15.2174					
7	−9.5459					

Table 3 Alg_3 against Alg_1 (literature)

	$\epsilon = 0.0001$											
Prob	Size		Cons		Alg_1				Alg_3			
	n_x	n_y	c_x	c_y	a	b	c	d	a	b	c	d
Ex1	5	2	3	3	4	2	1	0.452	1	2	1	0.361
Ex2	8	4	4	4	7	1	2	1.423	3	1	2	1.222
[2]	10	5	5	5	1	1	1	0.190	1	1	1	0.510
[10]	6	2	4	4	2	2	1	0.250	2	2	1	0.360
[13]	5	2	3	3	4	2	1	0.410	1	2	1	0.310
[13]×6	12	6	6	6	20	1	6	5.619	13	1	6	7.592
[13]×7	14	7	7	7	56	1	7	17.903	29	1	7	24.713
[13]×8	16	8	8	8	68	1	8	24.816	25	1	8	22.040
[13]×9	18	9	9	9	–	1	9	–	79	1	9	165.336
[13]×10	20	10	10	10	–	1	10	–	–	1	10	–
[21]	7	2	5	3	2	2	1	0.220	2	2	1	0.570

Prob : problem index,
Size : the number of variables in X_0 and Y_0,
Cons : the number of constraints in X_0 and Y_0,
a : the number of added cuts,
b : the iteration within which the global optimum is first touched,
c : the number of identified global optima,
d : solution time,
– : an unsolved problem when solution time exceeds 1,200 s

4 Computational experience

The two proposed enhanced cutting plane methods have been extensively tested against the two corresponding pure cutting plane methods using test problems from both the literature and from a class of applications. All experiments are conducted on a personal computer with Windows 2000, Matlab 6.5, Pentium-III 1,000 MHz CPU and 512 MB memory. The SQOPT is adopted to solve LP subproblems.

The test problems in Table 3 are taken from different references as indicated in the first column. It can be observed that for very small BLP problems, Alg_1 and Alg_3 are rather competitive, or sometimes Alg_3 is even inferior to Alg_1. The reason is that before computing the improved lower bound, we have to tighten the interval imposed on each variable. Each such tightening process needs to solve two LP programs. Correspondingly, the solution time saved from generating unnecessary cuts is balanced by the time for tightening these intervals, and therefore the observed computational results. However, as the size of a problem grows, the performance of Alg_1 and Alg_3 will be approaching and finally, Alg_3 will outperform Alg_1 by a very large factor. This effect can be observed from prob [13]×6 to prob [13] ×10 with

$$C^{m\times m} = \begin{bmatrix} -2 & 1 & 0 \dots & 0 & 0 \\ 1 & -2 & 1 \dots & 0 & 0 \\ 0 & 1 & -2 \ddots & 0 & 0 \\ \vdots & \vdots \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 \ddots & \ddots & 1 \\ 0 & 0 & 0 \dots & 1 & -2 \end{bmatrix},$$

$$A_1^{m\times m} = \begin{bmatrix} 1 & 2 & \dots & m-1 & m \\ 2 & 3 & \dots & m & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ m-1 & m & \dots & m-3 & m-2 \\ m & 1 & \dots & m-2 & m-1 \end{bmatrix} = A_2^{m\times m},$$

$$c^m = d^m = 0,$$
$$b_1^m = b_2^m = [m(m+1)/2, \dots, m(m+1)/2]^t,$$
$$x_1, \dots, x_m, y_1, \dots, y_m \geq 0.$$

Each problem has m local minimizers with equal objective values, and actually all of them are global minimizers. The computational load appears relatively heavy because the improved cutting plane algorithm cannot make much progress before all global optimizers are cut off. We observe that for prob [13]×6 and prob [13]×7, Alg_1 is even superior to Alg_3. Nevertheless, Alg_3 begins to outperform Alg_1 from prob [13]×8 even though neither of them can terminate within 1,200 s for prob [13]×10. The effect of having no knowledge about the global optimum becomes apparent due to the numerous cuts to be generated by Alg_1.

In Table 4, detailed computational results for the comparisons between Alg_2 and Alg_4 are provided. By comparing Table 4 with Table 3, we observe that for small size problems, Alg_2 is superior to Alg_3, but this is not the case for problems with larger sizes. For example, Alg_3 can solve prob [13]×9 within 165.336 s, while Alg_2 cannot solve the same problem within 1,200 s. We can also observe that Alg_2 and Alg_4 uniformly dominate Alg_1 and Alg_3, respectively. This fact indicates that EFIR and LOA_2

Table 4 Alg_4 against Alg_2 (literature)

	$\epsilon = 0.0001$											
Prob	Size		Cons		Alg_2				Alg_4			
	n_x	n_y	c_x	c_y	a	b	c	d	a	b	c	d
Ex1	5	2	3	3	3	1	1	0.190	1	1	1	0.170
Ex2	8	4	4	4	5	0	2	0.772	3	0	2	0.861
[2]	10	5	5	5	1	0	1	0.190	1	0	1	0.360
[10]	6	2	4	4	2	1	1	0.180	2	1	1	0.200
[13]	4	2	3	3	2	1	1	0.270	1	1	1	0.201
[13]×6	12	6	6	6	16	0	6	3.826	11	0	6	5.610
[13]×7	14	7	7	7	23	0	7	6.068	15	0	7	9.766
[13]×8	16	8	8	8	47	0	8	16.116	20	0	8	15.384
[13]×9	18	9	9	9	–	0	9	–	64	0	9	106.156
[13]×10	20	10	10	10	–	0	10	–	94	0	10	259.933
[21]	7	2	5	3	2	1	1	0.170	2	1	1	0.511

Prob : problem index,
Size : the number of variables in X_0 and Y_0,
Cons : the number of constraints in X_0 and Y_0,
a : the number of added cuts,
b : the iteration within which the global optimum is first touched,
c : the number of identified global optima,
d : solution time,
– : an unsolved problem when solution time exceeds 1,200 s

may significantly reduce the number of cuts to be generated even though we need to switch to LOA_1 when EFIR fails as more and more cuts are added, e.g., for prob [13]×7 through prob [13]×10. Therefore, Alg_4 tries to devote more computational effort in the early stages of our improved cutting plane method when EFIR is more successful and strive towards early termination when coupled with the underestimation routine.

The effectiveness of Alg_3 and Alg_4 can be further illustrated as the four cutting plane algorithms are applied to a special type of disjoint BLP programs arising in computational decision analysis [8], where

$$C^{2n\times 2n} = \begin{bmatrix} I_n & 0 \\ 0 & -I_n \end{bmatrix}, A_1^{m_1\times 2n}x \le b_1^{m_1}, A_2^{m_2\times 2n}y \le b_2^{m_2},$$
$$c^{2n} = d^{2n} = 0, 0 \le l_i^x \le x_i \le u_i^x \le 1, 0 \le l_i^y \le y_i \le u_i^y \le 1$$
$$\text{for } i = 1,\ldots,2n.$$

The variables $x_1,\ldots,x_{2n}$ and $y_1,\ldots,y_{2n}$ actually represent probability and utility variables, respectively. For a decision situation where the problem index is N, it has $2n$ x-variables and $2n$ y-variables, respectively. The number of linear constraints m_1 and m_2 are roughly equal to that of the variables in X_0 and Y_0, respectively. All these test problems for this class of applications were randomly generated.

In Table 5, the performance of Alg_3 and Alg_4 is quite encouraging in comparison with the two corresponding pure cutting plane algorithms that can solve only a small amount of test problems within 1,200 s. As for Alg_3 and Alg_4, it seems that the performance of Alg_3 is uniformly dominated by that of Alg_4 except for the group $N = 20$. In this group, we observe three out the ten test problems for which Alg_3 runs much faster than Alg_4. For example, one of them takes 24.312 s by using Alg_3, whereas the computing time rises to 66.876 s for Alg_4. In other groups, although this situation happens, the impact does not appear so strong. For over 90% of these problems, Alg_3 and Alg_4 terminated within five cuts, which indicates the lower bounding technique is relatively effective. Besides, EFIR and LOA_2 always take effect within this small

Table 5 Comparison between four algorithms (application)

$\epsilon = 0.0001$												
Prob	Size		Cons		Alg_1		Alg_2		Alg_3		Alg_4	
	n_x	n_y	c_x	c_y	n	t	n	t	n	t	n	t
N=15	60	30	15	14	4	–	5	–	10	4.751	10	4.210
N=20	80	40	22	20	3	–	3	–	10	6.224	10	12.174
N=25	100	50	26	24	3	–	3	–	10	9.027	10	5.160
N=30	120	60	31	30	4	–	5	–	10	11.264	10	7.880
N=35	140	70	35	34	2	–	4	–	10	15.902	10	12.959
N=40	160	80	42	40	1	–	1	–	10	21.165	10	13.499
N=45	180	90	46	44	–	–	–	–	10	27.867	10	18.367
N=50	200	100	51	50	–	–	–	–	10	29.273	10	19.846

Prob : problem index,
Size : the number of variables in X_0 and Y_0,
Cons : the number of constraints in X_0 and Y_0,
n : the number of solved problems,
t : average solution time for ten problems,
– : t >1,200 s for some problems in the group

number of generated cuts. EFIR fails for at most three instances in each group. The number of generated cuts when EFIR begins to fail ranges from 7 to 17. Of course this number is problem dependent. Consistent with our computational experience, most instances in computational decision analysis are well structured for the idea of seeking a WPGM by using EFIR and LOA_2 and incorporating the lower bounding technique to mitigate the need for an exhaustive search.

Acknowledgements The first author was supported by Fibre Science and Communication Network (FSCN). The second author was supported in part by the National Science Foundation Grant DMII-990826. The authors are grateful to two anonymous referees for recommendations that improved the clarity and presentation of the results.

References

1. Al-Khayyal, F.: Jointly constrained bilinear programs and related problems: an overview. Comput. Math. Appl. **19**(11), 53–62 (1990)
2. Al-Khayyal, F., Falk J.E.: Jointly constrained biconvex programming. Math. Oper. Res. **8**, 273–286 (1983)
3. Al-Khayyal, F., Kyparisis, J.: Finite convergence of algorithms for nonlinear programs and variational inequalities. J. Optim. Theory Appl. **70**, 319–332 (1991)
4. Alarie, S., Audet, C., Jaumard, B., Savard, G.: Concavity cuts for disjoint bilinear programming. Math. Program. **90**(2), 373–398 (2001)
5. Altman, M.: Bilinear programming. Bull D' Acad Pol. Des Sci. **16**(9), 741–746 (1968)
6. Audet, C., Hansen, P., Jaumard, B., Savard, G.: A symmetrical linear maxmin approach to disjoint bilinear programming. Math. Program. **85**(3), 573–592 (1999)
7. Balas, E.: Intersection cuts — a new type of cutting planes for integer programming. Oper. Res. **19**, 19–39 (1971)
8. Danielson, M., Ekenberg, L.: A framework for analyzing decisions under risk. Eur. J. Oper. Res. **104(3)**, 474–484 (1998)
9. Falk, J.E.: A linear max-min problem. Math. Program. **5**, 169–188 (1973)
10. Gallo, G., Ülkücü, A.: Bilinear programming: an exact algorithm. Math. Program. **12**, 173–194 (1977)
11. Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches, 3^{rd} edn. Springer-Verlag, Berlin (1996)
12. Júdice, J.J., Faustino, A.M.: Computational analysis of LCP methods for bilinear and concave quadratic programming. Comp. Oper. Res. **18(9)**, 645–654 (1991)
13. Konno, H.: A cutting plane algorithm for solving bilinear programs. Math. Program. **11**, 14–27 (1976a)
14. Konno, H.: Maximization of a convex quadratic function under linear constraints. Math. Program. **11**, 117–127 (1976b)
15. Majthay, M., Whinston, A.: Quasi-concave minimization subject to linear constraints. Discrete Math. **9**, 35–59 (1974)
16. Mangasarian, O.L.: Equilibrium points of bimatrix games. SIAM J. **12**, 778–780 (1964)
17. Mangasarian, O.L., Stone, H.: Two-person nonzero-sum games and quadratic programming. J. Math. Anal. Appl. **9**, 345–355 (1964)
18. Meyer, C.: A simple finite cone covering algorithm for concave minimization. J. Global Optim. **18(4)**, 357–365 (2000)
19. Shapiro, A., Al-Khayyal, F.: First-order conditions for isolated locally optimal solutions. J. Optim. Theory Appl. **77**, 189–196 (1993)
20. Sherali, H.D., Alameddine, A.: A new reformulation linearization algorithm for bilinear programming problems. J. of Global Optim. **2**, 379–410 (1992)
21. Sherali, H.D., Shetty, C.M.: A finitely convergent algorithm for bilinear programming problems using polar cuts and disjunctive face cuts. Math. Program. **19**, 14–31 (1980)
22. Soland, R.M.: Optimal facility location with concave costs. Oper. Res. **22**, 373–382 (1974)
23. Thieu, T.V.: A note on the solution of bilinear problems by reduction to concave minimization. Math. Program. **41**, 249–260 (1988)
24. Tuy, H.: Concave programming under linear constraints. Soviet Math. **5**, 1437–1440 (1964)

25. Tuy, H.: DC optimization: theory, methods and algorithms. In: Handbook of Global Optimization. Kluwer Academic Publishers, Dordrecht (1995)
26. Tuy, H.: Convex Analysis and Global Optimization. Kluwer Academic Publishers, Dotdrecht (1998)
27. Vaish, H., Shetty, C.M.: The bilinear programming problem. Math. Program. **23**, 303–309 (1976)
28. Vaish, H., Shetty, C.M.: A cutting plane algorithm for the bilinear programming problem. Naval Res. Logist. Quart. **24**, 83–94 (1977)
29. White, D.J.: A linear programming approach to solving bilinear programmes. Math. Program. **56**, 45–50 (1992)
30. Yajima, Y., Konno, H.: An efficient algorithm for solving rank two and rank three bilinear programming problems. J. Global Optim. **1**, 155–171 (1991)
31. Zwart, P.: Nonlinear programming: counterexamples to two global optimization algorithms. Oper. Res. **21(6)**, 1260–1266 (1973)