A New Reformulation-Linearization Technique for Bilinear Programming Problems*

HANIF D. SHERALI and AMINE ALAMEDDINE
Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061-0118, U.S.A.

(Received: 14 December 1990; accepted: 7 November 1991)

Abstract. This paper is concerned with the development of an algorithm for general bilinear programming problems. Such problems find numerous applications in economics and game theory, location theory, nonlinear multi-commodity network flows, dynamic assignment and production, and various risk management problems. The proposed approach develops a new Reformulation-Linearization Technique (RLT) for this problem, and imbeds it within a provably convergent branch-and-bound algorithm. The method first reformulates the problem by constructing a set of nonnegative variable factors using the problem constraints, and suitably multiplies combinations of these factors with the original problem constraints to generate additional valid nonlinear constraints. The resulting nonlinear program is subsequently linearized by defining a new set of variables, one for each nonlinear term. This "RLT" process yields a linear programming problem whose optimal value provides a tight lower bound on the optimal value to the bilinear programming problem. Various implementation schemes and constraint generation procedures are investigated for the purpose of further tightening the resulting linearization. The lower bound thus produced theoretically dominates, and practically is far tighter, than that obtained by using convex envelopes over hyper-rectangles. In fact, for some special cases, this process is shown to yield an exact linear programming representation. For the associated branch-and-bound algorithm, various admissible branching schemes are discussed, including one in which branching is performed by partitioning the intervals for only one set of variables x or y, whichever are fewer in number. Computational experience is provided to demonstrate the viability of the algorithm. For a large number of test problems from the literature, the initial bounding linear program itself solves the underlying bilinear programming problem.

Key words. Bilinear programming, nonconvex programming, global optimization, branch-and-bound, reformulation-linearization technique.

1. Introduction

This paper is concerned with the solution of bilinear programming problems of the form

$$BLP \text{ Minimize } \Phi(x, y) \equiv c^t x + d^t y + x^t G y \tag{1.1}$$

$$\text{subject to } (x, y) \in Z \equiv \left\{ (x, y): \begin{array}{ll} A_1 x + D_1 y \leq b_1 & (1.2a) \\ A_2 x + D_2 y = b_2 & (1.2b) \end{array} \right\}$$

$$(x, y) \in \Omega \equiv \{(x, y): 0 \leq l \leq x \leq u < \infty, 0 \leq L \leq y \leq U < \infty\}, \tag{1.3}$$

*This paper was presented at the II. IIASA Workshop on Global Optimization, Sopron (Hungary), December 9–14, 1990.

Journal of Global Optimization **2**: 379–410, 1992.

where $x \in R^n$, $y \in R^m$, Z is a polyhedron in R^{n+m} with designated inequality and equality constraints, and Ω is a hyper-rectangle in R^{n+m}. We assume the nontrivial case that $Z \cap \Omega \neq \phi$. Problems of this type find numerous applications in engineering, economics, and industrial environments including instances of nonlinear multi-commodity network flows, location-allocation problems with rectilinear distance metrics, finding Nash equilibrium points of a bimatrix game, dynamic Markovian assignment, multiple modular design problems, 3D assignment, dynamic production, risk management problems, and the track initialization problem. (See [1] for a list of references dealing with these applications.) Bilinear programming problems also arise in the solution of quadratic concave minimization problems [17], and linear complementarity problems [4]. Indeed, any linearly constrained quadratic programming problem with a general objective quadratic term z^tHz in some variable z can be put in form (1) by writing $z^tHz = z^ty$, and accommodating $y = Hz$ in the constraint set.

Among the known properties of the bilinear programming problem, is the fact that if x is fixed, then the problem becomes linear in y and *vice versa*. Also, the bilinear function x^tGy in the objective is nonconvex, and does not even enjoy any generalized convexity property such as quasi-convexity or quasi-concavity. Hence, bilinear programming problems admit the possibility of existence of several suboptimal local optima, which make even the simplest cases hard to solve.

In the form (1.1)–(1.3), the bilinear programming problem is said to be *jointly constrained* [2], and due to the bilinearity property, attains an optimum at a boundary point of $Z \cap \Omega$. However, several applications such as quadratic concave minimization, linear complementarity, location-allocation problems, bimatrix game theory problems, 3D-assignment problems, and the track initialization problem among others, admit cases in which the x and y variables are disjointly constrained. Such *separably constrained bilinear programs* [BLP(SC)] realize extreme point optimal solutions, if an optimum exists, despite the lack of quasi-concavity of the objective function.

Based on the extreme point optimality characterization and using suitable cutting planes, several algorithms have been developed for separably constrained bilinear programs. Pioneering work in this area was conducted by Konno [14, 15, 16, 17] who developed a cutting plane algorithm based on an enhancement of Tuy's [27] and Ritter's [20] methods. Using the theory of generalized polars, Vaish and Shetty [28] developed deeper cutting planes than those of Konno's class of cuts, and imbedded these in an infinitely convergent algorithm. These cutting planes were further improved by Sherali and Shetty [24] through the use of negative edge extensions of the simplicial cone incident at a given vertex, whenever the positive extension is contained within the polar set. By coordinating these polar cuts with suitable disjunctive face cuts, Sherali and Shetty developed an effective finitely convergent algorithm for BLP(SC).

Besides cutting plane methods, other algorithmic strategies have also been proposed for solving separably constrained bilinear programs. Vaish and Shetty

[28] developed a polyhedral annexation scheme in which a sequence of polytopes is inductively generated, for which the corresponding sequence of known optimal solutions converges finitely to a global optimum for BLP(SC). Another polyhedral annexation scheme has been proposed by Thieu [26], and is based on converting BLP(SC) to an equivalent concave minimization problem via a projection onto the space of x or y variables. Czochralska [8] employs standard linear programming techniques in conjunction with a method for ranking extreme points of polytopes in order to develop an alternative for this problem. Recently, Yajima and Konno [30] have suggested a novel parametric scheme for solving a special case of BLP(SC) in which the matrix G in (1.1) is of rank two or three. By exploiting this special structure, they are able to solve problems of size far larger than solvable using any available general purpose algorithm for BLP(SC).

A notable exception in the bilinear programming literature is the procedure by Al-Khayyal and Falk [2] (also see [5, 6]) that can be used to solve *jointly constrained* bilinear programs. Assuming that $G = I$ in (1.1), they develop an infinitely convergent branch-and-bound algorithm using lower bounds derived from convex envelopes of $x_i y_i$ for $i = 1, \ldots, n$, over rectangles in R^2 given by Ω in (1.3). When $G \neq I$, they suggest that the term x^tGy can be written as x^tz by accommodating the relationship $z = Gy$ in the constraint set, whence their method becomes applicable. The algorithm we propose can also be used to solve jointly constrained bilinear programming problems BLP of the form (1.1)–(1.3). (Actually, as in Al-Khayyal and Falk [2], we can solve biconvex programming problems in which the linear functions c^tx and d^ty in (11) are respectively replaced by convex functions $f(x)$ and $g(y)$. The only difference in the solution procedure is that this would lead to convex programming, rather than linear programming, lower bounding problems.) Our lower bounding linear program is obtained via a procedure that we call a Reformulation-Linearization Technique (RLT). This technique generates valid quadratic constraints that are subsequently linearized by defining new variables. The resulting linear program is shown to generate a lower bound that theoretically dominates that of Al-Khayyal and Falk [2], and that turns out computationally to be far tighter as well. Note, however, that in order to apply our RLT procedure, one does not need to transform x^tGy when $G \neq I$ as in Al-Khayyal and Falk. We imbed this lower bounding linear program into a branch-and-bound algorithm that is shown to converge infinitely to a global optimum for BLP. Moreover, our convergence proof suggests various alternative admissible branching schemes, including one in which the partitioning can be performed throughout on only the x or y-variables, for example, whichever is of lower dimension. Also, at each iteration, we can partition the problem based on either an x or a y variable to generate only two node sub-problems instead of the four nodes generated by Al-Khayyal and Falk. Computationally, this turns out to be a preferable strategy. Several of these admissible algorithmic strategies are computationally tested in order to propose an implementation scheme for our algorithm.

The remainder of this paper is organized as follows. Section 2 presents the fundamental RLT scheme. Section 3 imbeds the resulting lower bounding linear program in a branch-and-bound algorithm and establishes its convergence. Section 4 presents computational results on test problems from the literature as well as on pseudo-randomly generated problems. Section 5 demonstrates that the proposed RLT procedure actually generates the convex envelope representation of a bilinear function over special triangular and quadrilateral polytopes in R^2, hence motivating its use. Section 6 concludes with recommendations for implementation and some possible extensions.

Some specific notation used in the sequel is as follows. For a given real-valued, continuous function f defined on a convex set $X \subseteq R^n$, we denote its convex envelope over X as f_X (see, e.g. [12]). For a given optimization problem P, we denote its optimal objective value by $\nu[P]$. Finally, if $g_1(x) \geq 0$ and $g_2(x) \geq 0$ are some two inequality constraints numbered by equations (#) and (##), say, then by (#) · (##) we will mean the inequality $g_1(x)g_2(x) \geq 0$ obtained by algebraically multiplying the expressions for $g_1(x)$ and $g_2(x)$.

2. A New Reformulation-Linearization Technique (RLT)

In this section, we present the cornerstone of our methodology, namely, the Reformulation-Linearization Technique (RLT), that is used to generate tight linear programming based lower bounds. This procedure first generates valid quadratic constraints by using pairwise products of inequality constraints, or products of equality constraints with variables, and then subsequently linearizes these constraints by defining new variables. The bilinear objective term is similarly linearized, in order to produce a lower bounding linear program. Moreover, when the objective function is accommodated into the constraints in BLP as well as in this linear program through an auxiliary variable, the linear programming region projected onto the original variable space affords a tight polyhedral enveloping region for the closure of the convex hull of feasible points to the underlying bilinear program. In fact, as will be seen later, this polyhedral representation is exact in some special instances. We also remark here that a higher order variant of this scheme has been used by Sherali and Adams [21, 22] to generate a hierarchy of relaxations for linear pure and mixed-integer zero-one programming problems, spanning the spectrum from the linear programming relaxation to the (closure) convex hull of feasible solutions.

The fundamental steps of the RLT procedure are composed of two sequential phases – the reformulation phase, and the linearization phase. These are discussed below.

REFORMULATION PHASE

To apply the RLT scheme to problem BLP given by (1.1)–(1.3), define the following *bound factors* from (1.3):

$$(x_i - l_i) \geq 0, \quad i = 1, \ldots, n \tag{2.1a}$$

$$(u_i - x_i) \geq 0, \quad i = 1, \ldots, n \tag{2.1b}$$

$$(y_j - L_j) \geq 0, \quad j = 1, \ldots, m \tag{2.1c}$$

$$(U_j - y_j) \geq 0, \quad j = 1, \ldots, m \tag{2.1d}$$

Also, let b_1 be an R-vector and let A_{1r} and D_{1r} denote the rth rows of A_1 and D_1, respectively. Then the inequalities (1.2a) in Z can be expressed as the following *constraint factors*:

$$(b_{1r} - A_{1r}x - D_{1r}y) \geq 0, \quad r = 1, \ldots, R. \tag{2.1e}$$

In addition to the original constraints defining $Z \cap \Omega$, we now construct new implied nonlinear constraints by multiplying each of the constraints in (2.1) pairwise, including self products, in order to get various *classes of constraints*. As an illustration of such an operation, consider the multiplication of a set of bound factors with the set of inequality constraints in (2.1e):

$$(x_i - l_i)(b_{1r} - A_{1r}x - D_{1r}y) \geq 0 \quad \forall i, r.$$

Expanding this gives

$$-(A_{1r}x)x_i - (D_{1r}y)x_i + l_i(A_{1r}x) + l_i(D_{1r}y) + b_{1r}x_i \geq l_i b_{1r}$$
$$\text{for } i = 1, \ldots, n, \quad r = 1, \ldots, R.$$

As far as the equalities in (1.2b) are concerned, all that one needs to do is to multiply each of them with each x_i for all i and with each y_j for all j, in order to obtain the required class of equations. Then, if one were to consider an equality multiplied by any other linear expression or bound or constraint factor in (x, y), the resulting equation can be represented as a surrogate of the foregoing generated constraints, and so will be automatically implied by them. This is true even after executing the linearization step described subsequently below.

The original BLP constraints, together with constraints generated using the above pairwise product operations, yield a new equivalent reformulation of the bilinear programming problem as explicitly given below, where R and R' are respectively the number of inequality and equality constraints defining Z in (1.2).

$$BLP': \quad \text{Minimize} \quad \Phi(x, y) \equiv c^t x + d^t y + x^t G y$$

$$\text{subject to} \quad (x, y) \in Z \cap \Omega \tag{2.2a}$$

$$-(A_{1r}x)x_i - (D_{1r}y)x_i + l_i(A_{1r}x) + l_i(D_{1r}y) + b_{1r}x_i \geq l_i b_{1r},$$
$$i = 1, \ldots, n, \quad r = 1, \ldots, R \tag{2.2b}$$

$$(A_{1r}x)x_i + (D_{1r}y)x_i - u_i(A_{1r}x) - u_i(D_{1r}y) - b_{1r}x_i \geq -u_i b_{1r},$$
$$i = 1, \ldots, n, \quad r = 1, \ldots, R \tag{2.2c}$$

$$-(A_{1r}x)y_j - (D_{1r}y)y_j + L_j(A_{1r}x) + L_j(D_{1r}y) + b_{1r}y_j \geq L_j b_{1r},$$
$$j = 1 \ldots, m, \quad r = 1, \ldots, R \tag{2.2d}$$

$$(A_{1r}x)y_j + (D_{1r}y)y_j - U_j(A_{1r}x) - U_j(D_{1r}y) - b_{1r}y_j \geq -U_j b_{1r}, \quad j = 1, \ldots, m, \quad r = 1, \ldots, R \tag{2.2e}$$

$$x_i x_j - l_j x_i - l_i x_j \geq l_i l_j, \ \forall 1 \leq i \leq j \leq n, \tag{2.2f}$$

$$y_i y_j - L_j y_i - L_i y_j \geq -L_i L_j, \ \forall 1 \leq i \leq j \leq m, \tag{2.2g}$$

$$x_i x_j - u_j x_i - u_i x_j \geq -u_i u_j, \ \forall 1 \leq i \leq j \leq n, \tag{2.2h}$$

$$y_i y_j - U_j y_j - U_i y_j \geq -U_i U_j, \ \forall 1 \leq i \leq j \leq m, \tag{2.2i}$$

$$x_i y_j - L_j x_i - l_i y_j \geq -l_i L_j, \ \forall i = 1, \ldots, n, \ j = 1, \ldots, m, \tag{2.2j}$$

$$x_i y_j - U_j x_i - u_i y_j \geq -u_i U_j, \ \forall i = 1, \ldots, n, \ j = 1, \ldots, m, \tag{2.2k}$$

$$-x_i y_j + U_j x_i + l_i y_j \geq l_i U_j, \ \forall i = 1, \ldots, n, \ j = 1, \ldots, m, \tag{2.2l}$$

$$-x_i y_j + L_j x_i + u_i y_j \geq u_i L_j, \ \forall i = 1, \ldots, n, \ j = 1, \ldots, m, \tag{2.2m}$$

$$-x_i x_j + u_j x_i + l_i x_j \geq l_i u_j, \ \forall i = 1, \ldots, n, \ j = 1, \ldots, n, \tag{2.2n}$$

$$-y_i y_j + U_j y_i + L_i y_j \geq L_i U_j, \ \forall i = 1, \ldots, m, \ j = 1, \ldots, m, \tag{2.2o}$$

$$x^t[A^t_{1i}D_{1j} + A^t_{1j}D_{1i}]y + x^t A^t_{1i}A_{1j}x + y^t D^t_{1i}D_{1j}y - (b_{1i}A_{1j} + b_{1j}A_{1i})x - (b_{1i}D_{1j} + b_{1j}D_{1i})y + b_{1i}b_{1j} \geq 0, \ \forall 1 \leq i \leq j \leq R \tag{2.2p}$$

$$(A_{2r}x)x_i + (D_{2r}y)x_i - b_{2r}x_i = 0, \ i = 1, \ldots, n, \ r = 1, \ldots, R' \tag{2.2q}$$

$$(A_{2r}x)y_j + (D_{2r}y)y_j - b_{2r}y_j = 0, \ j = 1, \ldots, m, \ r = 1, \ldots, R'. \tag{2.2r}$$

The restrictions given in (2.2b)–(2.2r) are all valid implied nonlinear constraints. Note that any suitable interproducts of the bound and constraint factors, constructed as above, can be used to generate a *reformulation* of the BLP. Hence, such a reformulation can contain not only all possible products of (2.1) taken two a time, but can also include higher order products. Of course, the more the constraints included of this type, the stronger might be the resulting representation obtained, but also, the larger the size of this representation. Hence, from a computational viewpoint, a suitable compromise needs to be struck.

LINEARIZATION PHASE

The linearization is achieved through an appropriate *variable substitution strategy*, which transforms the generated set of nonlinear constraints ((2.2b)–(2.2r)), to a set of linear constraints. Specifically, we substitute

$$\begin{aligned} w_{ij} &= x_i y_j \quad \text{for all} \quad i = 1, \ldots, n, \quad j = 1, \ldots, m, \\ X_{ij} &= x_i x_j \quad \text{for all} \quad 1 \leq i \leq j \leq n, \\ Y_{ij} &= y_i y_j \quad \text{for all} \quad 1 \leq i \leq j \leq m. \end{aligned} \tag{2.3}$$

This linearizes the reformulated nonlinear problem to the form

$$\text{LP}(\Omega)\text{: Minimize } \Phi_L(x, y, w) \equiv c^t x + d^t y + \sum_{i=1}^{n} \sum_{j=1}^{m} g_{ij} w_{ij}$$
$$\text{subject to } (x, y, w, X, Y) \in Z_L \cap \Omega_L \,, \tag{2.4}$$

where $G = [g_{ij}]$, Z_L is the linearized set of constraints (2.2) under the transformation (2.3), and where

$$\begin{aligned} \Omega_L = \{(x, y, w, X, Y)\colon (x, y) \in \Omega \,,\ & l_i L_j \leq w_{ij} \leq u_i U_j \,, \\ & \text{for } i = 1, \ldots, n,\ j = 1, \ldots, m \\ & l_i l_j \leq X_{ij} \leq u_i u_j \,, \\ & \text{for } 1 \leq i \leq j \leq n \,, \\ & L_i L_j \leq Y_{ij} \leq U_i U_j \,, \\ & \text{for } 1 \leq i \leq j \leq m\} \end{aligned} \tag{2.5}$$

Note that the bounds on the w, X, and Y variables in Ω_L are implied by the constraints in Z_L, and are included here only for convenience when some constraints in Z_L are later relaxed. Also, the terminology LP(Ω) is used to emphasize that if the variable bounds defining Ω are modified by some partitioning process, then the formulation of the corresponding problem (2.4) changes accordingly. In a likewise manner, let us call BLP as BLP(Ω).

Note that LP(Ω) is a relaxation of BLP(Ω) in the sense that for any solution $(\bar{x}, \bar{y})$ feasible to the latter problem, there exist $\bar{w}$, $\bar{X}$, $\bar{Y}$, constructed via the substitution (2.3), such that, $(\bar{x}, \bar{y}, \bar{w}, \bar{X}, \bar{Y})$ is a feasible solution to the former problem, with the same objective value. However, the converse is not necessarily true, and so, LP(Ω) yields a lower bound for BLP(Ω). However, if an optimal solution to LP(Ω) automatically satisfies (2.3), then this solution evidently solves BLP(Ω) as well. More generally, if an optimal solution $(\bar{x}, \bar{y}, \bar{w}, \bar{X}, \bar{Y})$ to LP(Ω) yields

$$\sum_{i=1}^{n} \sum_{j=1}^{m} g_{ij} \bar{w}_{ij} = \bar{x}^t G \bar{y} \,, \tag{2.6}$$

then $(\bar{x}, \bar{y})$ solves BLP(Ω) since there exists an (alternative) optimal solution to LP(Ω) with $x = \bar{x}$, $y = \bar{y}$, and with $(\bar{w}, \bar{X}, \bar{Y})$ satisfying (2.3). This statement is formally summarized below. Henceforth, let $v[\cdot]$ be the value at optimality of problem $[\cdot]$.

LEMMA 1. *$v[LP(\Omega)] \leq v[BLP(\Omega)]$. Moreover, if $(\bar{x}, \bar{y}, \bar{w}, \bar{X}, \bar{Y})$ solves $LP(\Omega)$ and if $\sum_{i=1}^{n} \sum_{j=1}^{m} g_{ij} \bar{w}_{ij} = \bar{x}^t G \bar{y}$, then $(\bar{x}, \bar{y})$ solves $BLP(\Omega)$.*

REMARK 1. The set Z_L can have far too many constraints to be unarguably advisable in all instances. Hence, the question to be addressed is which constraints in Z_L should be constructed and which dispensed. Note that the objective in (1.1) has terms $g_{ij} x_i y_j$, and so if $g_{ij} > 0$, the cross products $x_i y_j$ are diminished by minimizing the objective, and therefore, we need implied constraints that

generate lower bounds on w_{ij}. Similarly, constraints that provide upper bounds on w_{ij} are needed if $g_{ij} < 0$. Hence, for example, we can generate constraints based on the following products to be included in Z_L, where $\cdot$ denotes the constraint product operation.

$$[(2.1a)\cdot(2.1c) \text{ and } (2.1b)\cdot(2.1d), \text{ if } g_{ij} \geq 0], \tag{2.7a}$$

$$[(2.1a)\cdot(2.1d) \text{ and } (2.1b)\cdot(2.1c), \text{ if } g_{ij} < 0]. \tag{2.7b}$$

Actually, these constraints can be verified to correspond respectively to convex and concave envelopes of $x_i y_j$ over $l_i \leq x_i \leq u_i$, and $L_j \leq y_j \leq U_j$. In fact, the following result holds. Here, for a given function $[\cdot]$, we denote by $[\cdot]_\Omega$ its convex envelope over the set Ω.

THEOREM 1. *Consider LP'(Ω) obtained from LP(Ω) by relaxing Z_L to Z_L', where Z_L' represents the constraints in Z and those linearized from* (2.7), *and by relaxing Ω_L to Ω. Then,*

$$\nu[\mathrm{LP}'(\Omega)] \equiv \text{minimum} \left\{ c^t x + d^t y + \sum_{i=1}^{n} \sum_{j=1}^{m} [g_{ij} x_i y_j]_\Omega : (x, y) \in Z \cap \Omega \right\} \tag{2.8}$$

Proof. By definition, we have $\nu[\mathrm{LP}'(\Omega)] = \text{minimum } \{c^t x + d^t y + \Sigma_i \Sigma_j g_{ij} w_{ij} : (x, y) \in Z \cap \Omega, (x, y, w) \in Z^+ \cap Z^-\}$, where

$$Z^+ \equiv \{(x, y, w): w_{ij} \geq L_j x_i + l_i y_j - l_i L_j, \; w_{ij} \geq U_j x_i + u_i y_j - u_i U_j, \; \forall i, j\} \tag{2.9a}$$

and

$$Z^- \equiv \{(x, y, w): w_{ij} \leq U_j x_i + l_i y_j - l_i U_j, \; w_{ij} \leq L_j x_i + u_i y_j - u_i L_j, \; \forall i, j\} \tag{2.9b}$$

are the linearized forms of (2.7a) and (2.7b), respectively. Thus,

$$\begin{aligned} \nu[\mathrm{LP}'(\Omega)] = \text{minimum } \{ & c^t x + d^t y + \sum_{(ij):\, g_{ij} \geq 0} \sum g_{ij} \max \{L_j x_i + l_i y_j - l_i L_j, \\ & U_j x_i + u_i y_j - u_i U_j\} + \sum_{(ij):\, g_{ij} < 0} \sum g_{ij} \min \{U_j x_i + l_i y_j - l_i U_j, \\ & L_j x_i + u_i y_j - u_i L_j\} : (x, y) \in Z \cap \Omega \}. \end{aligned} \tag{2.10}$$

Now, recall from the definition of the convex envelope of $x_i y_j$ over a rectangular region in R^2 (see Al-Khayyal and Falk [2] and Al-Khayyal [6]) that

$$\begin{aligned} [g_{ij} x_i y_j]_\Omega = g_{ij} [x_i y_j]_\Omega = g_{ij} \max \{ & L_j x_i + l_i y_j - l_i L_j, \\ & U_j x_i + u_i y_j - u_i U_j\}, \quad \text{if } g_{ij} \geq 0, \end{aligned} \tag{2.11a}$$

and

$$[g_{ij} x_i y_j]_\Omega = g_{ij} \min \{U_j x_i + l_i y_j - l_i U_j, \; L_j x_i + u_i y_j - u_i L_j\}, \quad \text{if } g_{ij} < 0. \tag{2.11b}$$

Consequently, (2.10) and (2.11) imply that (2.8) holds true, and this completes the proof. □

COROLLARY 1. *$v[LP(\Omega)]$ exceeds Al-Khayyal and Falk's lower bound for BLP (Ω).*

Proof. Al-Khayyal and Falk's [2] lower bound for BLP(Ω) when extended to the case $G \neq I$, is precisely given by (2.8). Since $v[\text{LP}(\Omega)] \geq v[\text{LP}'(\Omega)]$, because LP'($\Omega$) is a relaxation of LP($\Omega$), the result follows. □

DESIGN OF A SUITABLE RLT SCHEME

In light of the foregoing discussion on formulating a suitable linear bounding problem, we briefly suggest certain alternatives that can be investigated.

Strategies

1. Suppose that we solve the linear program LP'(Ω), defined in Theorem 1, and obtain an optimum $(\bar{x}, \bar{y}, \bar{w})$. Then *fixing* $(x, y, w) \equiv (\bar{x}, \bar{y}, \bar{w})$, and subject to $(x, y, w, X, Y) \in \Omega_L$, we can test which of the other constraints in Z_L are individually unsatisfiable, or for which the maximum slack value (for the inequalities) is lesser than some threshold value, and then generate and include these constraints in the problem.
2. Alternatively, we can also fix $X_{ij} = \bar{X}_{ij} \equiv \bar{x}_i \bar{x}_j$, for all $1 \leq i \leq j \leq n$ and $Y_{ij} = \bar{Y}_{ij} \equiv \bar{y}_i \bar{y}_j$, for all $1 \leq i \leq j \leq m$, and test the feasibility of $(\bar{x}, \bar{y}, \bar{w}, \bar{X}, \bar{Y})$ with respect to the individual constraints of Z_L in order to generate a suitable subset that appears to tighten the linear programming relaxation.
3. Also, certain manageable subsets of constraints can be identified with respect to which all possible second or higher order factor products are generated.

In our computational experiments, we attempted strategy 2 (see Remark 7 for details), and we recommend the investigation of other such strategies for future research.

3. A Branch-and-Bound Algorithm

Having selected some suitable RLT scheme, we can imbed the corresponding lower bounding linear program in a branch-and-bound algorithm where the partitioning is performed by decomposing Ω into sub-hyper-rectangles. Hence, at each *stage* k, we have a set of active nodes (k, t) for t in some index set T_k. Associated with each node (k, t) of the branch-and-bound tree, we have a hyper-rectangle $\Omega^{(k,t)}$ replacing Ω, with the corresponding node subproblem being the bilinear program BLP[$\Omega^{(k,t)}$], and the corresponding lower bounding linear programming problem being LP[$\Omega^{(k,t)}$]. To begin with, when $k = 0$, we have $T_0 = \{1\}$, and $\Omega^{(0,1)} \equiv \Omega$. Thereafter, at each subsequent stage, having selected an

active node, we partition it into two subnodes, and analyze these two new nodes, in order to derive the set of active nodes at the subsequent stage.

Below, we give a theorem that is central to the branching or partitioning process. Although this is stated relative to the original bilinear programming problem, it is applicable at any node subproblem as well.

THEOREM 2. *Assume that LP(Ω) includes the constraints generated via the products in* (2.7) *and let* $(\bar{x}, \bar{y}, \bar{w}, \bar{X}, \bar{Y})$ *be an optimal solution. Determine*

$$(p, q) \in \underset{(i,j)}{\operatorname{argmax}} [g_{ij}(\bar{x}_i\bar{y}_j - \bar{w}_{ij})] . \tag{3.1a}$$

Then, if $\Phi_L(\bar{x}, \bar{y}, \bar{w}) < \Phi(\bar{x}, \bar{y})$, *we have,*

$$l_p < \bar{x}_p < u_p , \quad \text{and} \quad L_q < \bar{y}_p < U_q . \tag{3.1b}$$

Proof. For any i, j, one has the following constraints included in LP(Ω),

$$w_{ij} \geq L_j x_i + l_i y_j - l_i L_j , \tag{3.2a}$$

$$w_{ij} \geq U_j x_i + u_i y_j - u_i U_j , \tag{3.2b}$$

$$w_{ij} \leq U_j x_i + l_i y_j - l_i U_j , \tag{3.2c}$$

$$w_{ij} \leq L_j x_i + u_i y_j - u_i L_j . \tag{3.2d}$$

Now, suppose that $\bar{x}_i = l_i$. Then (3.2a) implies that $\bar{w}_{ij} \geq \bar{y}_j l_i$ and also (3.2c) implies that $\bar{w}_{ij} \leq \bar{y}_j l_i$ which gives $\bar{w}_{ij} = \bar{y}_j l_i \equiv \bar{x}_i \bar{y}_j$. Similarly, $\bar{w}_{ij} \equiv \bar{x}_i \bar{y}_j$ if $\bar{x}_i = u_i$ or $\bar{y}_j = L_j$ or $\bar{y}_j = U_j$. Consequently, since $\Phi_L(\bar{x}, \bar{y}, \bar{w}) < \Phi(\bar{x}, \bar{y})$ implies that

$$\sum_i \sum_j g_{ij}\bar{w}_{ij} < \sum_i \sum_j g_{ij}\bar{x}_i\bar{y}_j ,$$

this yields $g_{pq}\bar{w}_{pq} < g_{pq}\bar{x}_p\bar{y}_q$ where (p, q) is given by (3.1a). Therefore from above, this in turn implies that (3.1b) holds, and the proof is complete. □

Theorem 2 leads to the following *Branching Rule* as applied at node zero. Naturally this is symmetrically applicable at all other nodes as well.

BRANCHING VARIABLE SELECTION AND PARTITIONING STRATEGIES

Branching Rule 1: Find

$$(p, q) \in \underset{(i,j)}{\operatorname{argmax}} [g_{ij}(\bar{x}_i\bar{y}_j - \bar{w}_{ij})] \tag{3.3a}$$

and compute

$$\nu_p = \sum_{j=1}^{m} [g_{pj}(\bar{x}_p\bar{y}_j - \bar{w}_{pj})] , \; V_q = \sum_{i=1}^{n} [g_{iq}(\bar{x}_i\bar{y}_q - \bar{w}_{iq})] . \tag{3.3b}$$

If $\nu_p \geq V_q$,

$$\text{partition } l_p \leq x_p \leq u_p \text{ into } l_p \leq x_p \leq \bar{x}_p \text{ and } \bar{x}_p \leq x_p \leq u_p . \tag{3.3c}$$

Otherwise,

$$\text{partition } L_q \leq y_q \leq U_q \text{ into } L_q \leq y_q \leq \bar{y}_q \text{ and } \bar{y}_q \leq y_q \leq U_q . \tag{3.3d}$$

Branching Rule 2. As an alternative strategy, we can use absolute values in (3.3b) and compute

$$\nu_p = \sum_{j=1}^{m} |g_{pj}(\bar{x}_p \bar{y}_j - \bar{w}_{pj})| , \; V_q = \sum_{i=1}^{n} |g_{iq}(\bar{x}_i \bar{y}_q - \bar{w}_{iq})| . \tag{3.3b}'$$

and then adopt (3.3c) and (3.3d).

We note here that whereas the underlying motivation for employing (3.3b) is guided by the contribution of each variable to the gap between the lower and upper bounds on the objective function value, the reason behind incorporating the absolute value operator as an alternative in (3.3b)′, is to base the partitioning on the absolute discrepancy between the relevant components of w and the corresponding terms in $x^t y$ as dictated by feasibility to BLP′.

Other branching rules are also possible. As an example, and as a consequence of the convergence theorem (given later), branching could be made on the set of x or y-variables alone, instead of on both x and y sets of variables. This along with another rule are presented after the Convergence Theorem (Theorem 3). But first we present the schema of our algorithm.

OUTLINE OF THE ALGORITHM

Initialization Step. Initialize $\Omega^{01} = \Omega$ and let $T_0 = \{1\}$ index the single node corresponding to $\Omega = \Omega^{01}$ (henceforth referred to as *node* Ω^{01}) at stage zero of the branch-and-bound tree. Solve LP(Ω^{01}) to obtain the partial solution (x^0, y^0, w^0). If $\Phi_L(x^0, y^0, w^0) = \Phi(x^0, y^0)$ terminate with (x^0, y^0) as an optimal solution to BLP(Ω). Otherwise, let $\text{LB}_0 = \text{LB}_{01} = \Phi_L(x^0, y^0, w^0)$, and $\text{UB}_0 = \text{UB}_{01} = \Phi(x^0, y^0)$, be the current lower and upper bounds, and determine a branching variable for the node Ω^{01} using (3.3) with $(\bar{x}, \bar{y}, \bar{w}) = (x^0, y^0, w^0)$. Put $k = 1$, let the selected node at stage zero be $t^* = 1$, and go to the Main Step.

Main Step (*Stage* k). Partition the node $\Omega^{(k-1)t^*}$, where $t^* \in T_{k-1}$, into two subproblem nodes according to (3.3). Call these nodes Ω^{k1} and Ω^{k2}. Let the nodes at stage k be

$$\{\Omega^{kt}, t \in T_k\} \equiv \{\Omega^{k1}, \Omega^{k2}\} \cup \{\Omega^{(k-1)t}, t \in T_{k-1}, t \neq t^*\} .$$

For each $t = 1, 2$ solve LP(Ω^{kt}) and let the corresponding (partial) solution obtained be denoted by (x^{kt}, y^{kt}, w^{kt}). Put $\text{LB}_{kt} = \Phi_L(x^{kt}, y^{kt}, w^{kt})$, $\text{UB}_{kt} = \Phi(x^{kt}, y^{kt})$, and use (3.3) to select and store the respective branching variables to be

used for these nodes, when and if they are partitioned. (Note that the similar entities (x^{kt}, y^{kt}, w^{kt}), LB_{kt}, UB_{kt}, and the branching variable choices, are known for the other nodes $\Omega^{(kt)}$, $t \in T_k$, $t > 2$, from stage $k-1$.)

Updating Operations

1. Compute the stage k upper bound as

$$\mathrm{UB}_k = \min \{\mathrm{UB}_{(k-1)}, \mathrm{UB}_{k1}, \mathrm{UB}_{k2}\}, \text{ and let } (x^k, y^k) \tag{3.4a}$$

be the corresponding incumbent for which $\mathrm{UB}_k = \Phi(x^k, y^k)$.

2. Compute the stage k lower bound along with the index $t^* \in T_k$ as

$$\mathrm{LB}_k = \mathrm{LB}_{kt^*}, \text{ where } t^* \in \operatorname{argmin} \{\mathrm{LB}_{kt}, t \in T_k\}. \tag{3.4b}$$

3. Accordingly, update the active set of nodes at stage k as

$$T_k \leftarrow T_k - \{t \in T_k : \mathrm{LB}_{kt} \geq \mathrm{UB}_k\}. \tag{3.4c}$$

Termination check. If $T_k = \phi$, then STOP; the current incumbent solution is optimal for BLP. Otherwise, increment k by one and return to the Main Step.

REMARK 2. Practically, we can employ a given termination tolerance $\varepsilon > 0$ in (3.3c) for fathoming only marginally improving nodes by replacing T_k with $T_k - \{t \in T_k : \mathrm{LB}_{kt} + \varepsilon \geq \mathrm{UB}_k\}$. Specifically, we suggest using

$$\varepsilon = \mathrm{Max} \{\varepsilon_1, [\mathrm{UB}_0 - \mathrm{LB}_0]\varepsilon_2\}$$

where $\varepsilon_1 > 0$ and $\varepsilon_2 > 0$ are some small (0.001–0.01) user specified tolerances.

CONVERGENCE ANALYSIS

We now establish the convergence of the above algorithm. Al-Khayyal and Falk's [2] algorithm is based on the BLP formulation in Remark 1. They perform 4-way rectangular partitions, and use the equicontinuity argument of Horst [10]. Horst's 1986 paper is an updated version of his 1976 paper where he bases his convergence argument on the same node selection strategy as we use in this research, but assumes a consistency property whose verification in special cases is the major burden of the proof. Our proof has the same skeletal framework as Horst's [11], but its content is different, and the equivalent step in the proof of verifying the consistency property actually reveals different convergent variants of the algorithm. We also note that our choice of iterates which establish the convergence argument is different, since we choose these iterates in a specific manner based on a given nested partition. Additionally, our convergence theorem shows that the partitioning can be performed to create two subnodes based on current iterate values, and can be readily extended to affirmatively answer the open convergence question posed by Al-Khayyal and Larsen [7]. Moreover, since we create only

two new nodes instead of four as done by Al-Khayyal and Falk [2], we show in the sequel that convergence is guaranteed even if we partition on only one of the sets of variables x or y. This may be computationally advantageous when n is much smaller than m, or *vice versa*. We also remark here that as evident from the proof of Theorem 3 below, the foregoing algorithmic modifications are valid even for Al-Khayyal and Falk's [2] procedure which uses convex envelopes over hyper-rectangles at the lower bounding step.

Before we state and prove the convergence theorem, we need the following lemma because of the peculiar nature of RLT.

LEMMA 2. *At any stage k, let the node Ω^{kt^*} be split into nodes $\Omega^{(k+1),1}$ and $\Omega^{(k+1),2}$ at stage $(k+1)$. Then,*

$$\min\{\mathrm{LB}_{(k+1),1}, \mathrm{LB}_{(k+1),2}\} \geq \mathrm{LB}_{kt^*}.$$

Proof. Let us show that $\mathrm{LB}_{(k+1),1} \geq \mathrm{LB}_{kt^*}$ by showing that $\mathrm{LP}_R(\Omega^{(k+1),1}) \subseteq \mathrm{LP}_R(\Omega^{kt^*})$, where $\mathrm{LP}_R(\cdot)$ denotes the feasible region of $\mathrm{LP}(\cdot)$ in the linearized (x, y, w, X, Y)-space. The case of $\mathrm{LB}_{(k+1),2} \geq \mathrm{LB}_{kt^*}$ is symmetric. Hence, without loss of generality, suppose that $\Omega^{(k+1),1}$ is obtained from Ω^{kt^*} by modifying $x_p \leq u_p$ to $x_p \leq u'_p$, say, where $u'_p < u_p$. (The argument for modifying a lower bound is identical.) Now, consider any $(\bar{x}, \bar{y}, \bar{w}, \bar{X}, \bar{Y}) \in \mathrm{LP}_R(\Omega^{(k+1),1})$. Note that the constraints of $\mathrm{LP}_R(\Omega^{(k+1),1})$ and $\mathrm{LP}_R(\Omega^{kt^*})$ are identical, except for those constraints involving the upper bound on x_p. For these constraints, since $\bar{x}_p \leq u'_p$ and $u'_p < u_p$, we have $\bar{x}_p \leq u_p$. Furthermore, for any original problem constraint $\alpha^t x + \beta^t y \geq \gamma$, the product with the factor $(u'_p - x_p)$ yields the linearized constraint

$$u'_p(\alpha^t x + \beta^t y - \gamma) \geq (\alpha^t X_p + \beta^t w_p - \gamma x_p)$$

where $X_p \equiv (x_p)x$ and $w_p \equiv (x_p)y$. Since $(\alpha^t \bar{x} + \beta^t \bar{y} - \gamma) \geq 0$ and $u_p > u'_p$, we have,

$$u_p(\alpha^t \bar{x} + \beta^t \bar{y} - \gamma) \geq u'_p(\alpha^t \bar{x} + \beta^t \bar{y} - \gamma) \geq (\alpha^t \bar{X}_p + \beta^t \bar{w}_p - \gamma \bar{x}_p).$$

Therefore, $(\bar{x}, \bar{y}, \bar{w}, \bar{X}, \bar{Y}) \in \mathrm{LP}_R(\Omega^{kt^*})$ as well, and this completes the proof. □

THEOREM 3. (*Main Convergence Result*). *The proposed branch-and-bound algorithm either terminates finitely with an optimal solution to BLP(Ω), or else, it generates a sequence of iterates $\{(x^k, y^k)\}$, along with a sequence of nonincreasing upper bounds $\{UB_k\}$ and a sequence of nondecreasing lower bounds $\{LB_k\}$, such that*

$$\lim \mathrm{LB}_k = \overline{\mathrm{LB}} = \lim \mathrm{UB}_k = \overline{\mathrm{UB}} = z^* \equiv \nu[\mathrm{BLP}(\Omega)].$$

Proof. By Lemma 1 and Theorem 2, the algorithm is well defined, and if it terminates finitely, the solution (x^k, y^k) is optimal to problem BLP(Ω). Hence,

suppose that an infinite sequence is generated. Clearly, $\{\mathrm{UB}_k\}$ is nonincreasing. Furthermore, $\{\mathrm{LB}_k\}$ is nondecreasing by Lemma 2.

Now, since an infinite sequence is generated, some branch in the enumeration tree is comprised of an infinite number of descendants. Associated with this branch is a *nested sequence* of partitions

$$\Omega^{kt(k)},\ t(k) \in T_k,\ \forall \in K, \text{ satisfying } \Omega^{k't(k')} \subset \Omega^{kt(k)}$$

$$\text{for } k' = \min\{k'' \in K:\ k'' > k\},\quad \forall k \in K,$$

where K is an index subsequence of $0, 1, 2, \ldots$, chosen suitably as follows. Note that a node can recur for several stages. Hence, the particular $k \in K$ used for which $\Omega^{kt(k)}$ identifies the partition at a given node in the nested sequence, is that at which this partition is branched on for the next stage. Accordingly, $t(k)$ corresponds to the particular $t^* \in T_k$. Hence, by (3.4b) and the main step of the algorithm, $\mathrm{LB}_k \equiv \mathrm{LB}_{kt(k)}$, $\forall k \in K$.

Consequently, we have,

$$\Phi(x^{kt(k)}, y^{kt(k)}) - \Phi_L(x^{kt(k)}, y^{kt(k)}, w^{kt(k)}) \geq \mathrm{UB}_k - \mathrm{LB}_k > 0,\quad \forall k \in K. \tag{3.5}$$

Now, because $\Omega^{kt(k)}$, $k \in K$, is a nested sequence, there exists some variable, say, x_p without loss of generality, that is selected as the branching variable, and has its interval split infinitely often. Accompanying the splitting of the x_p interval, there must be some y_q in (3.3) which occurs infinitely often. Call the index set of iterates in K when (p, q) occurs in (3.3a) as $K_1 \subseteq K$. Hence, from (3.3a), we get

$$g_{ij}[x_i^{kt(k)}\, y_j^{kt(k)} - w_{ij}^{kt(k)}] \leq g_{pq}[x_p^{kt(k)}\, y_q^{kt(k)} - w_{pq}^{kt(k)}]$$

$\forall (i, j)$, and $\forall k \in K_1$. Summing up over all (i, j) one gets using (1.1), (2.4), and (3.5) that,

$$\begin{aligned} 0 < \mathrm{UB}_k - \mathrm{LB}_k &\leq \Phi(x^{kt(k)}, y^{kt(k)}) - \Phi_L(x^{kt(k)}, y^{kt(k)}, w^{kt(k)}) \\ &\leq mn\, g_{pq}[x_p^{kt(k)}\, y_q^{kt(k)} - w_{pq}^{kt(k)}],\quad \forall k \in K_1. \end{aligned} \tag{3.6}$$

Next, let $[l_p^{kt(k)}, u_p^{kt(k)}]$ and $[L_q^{kt(k)}, U_q^{kt(k)}]$ represent the intervals for x_p and y_q, respectively, in the sequence $\{\Omega^{kt(k)}\}$. Since the sequence

$$\{x^{kt(k)}, y^{kt(k)}, w^{kt(k)}, l_p^{kt(k)}, u_p^{kt(k)}, L_q^{kt(k)}, U_q^{kt(k)}\}_{K_1}$$

is bounded, there exists a convergent subsequence indexed by $K_2 \subseteq K_1$, with limit point

$$(\bar{x}, \bar{y}, \bar{w}, \bar{l}_p, \bar{u}_p, \bar{L}_q, \bar{U}_q).$$

Because of the nested nature of $\{\Omega^{kt(k)}\}$, $k \in K_2$, and by the partitioning in (3.3c) and (3.3d), we know that for each $k \in K_2$, $x_p^{kt(k)} \notin (l_p^{k't(k')}, u_p^{k't(k')})$, $\forall k' \in K_2$, $k' > k$. But $x_p^{kt(k)} \in [l_p^{kt(k)}, u_p^{kt(k)}]$, $\forall k \in K_2$, and so, $\bar{x}_p \in [\bar{l}_p, \bar{u}_p]$. This implies that $\bar{x}_p = \bar{l}_p$ or $\bar{x}_p = \bar{u}_p$, since otherwise, we would have a contradiction with the foregoing statement. Following the proof of Theorem 2 over the iterates $k \in K_2$,

we see that this in turn implies that $\bar{w}_{pq} = \bar{x}_p \bar{y}_q$. Hence, taking limits in (3.6) as $k \to \infty$, $k \in K_2 \subseteq K_1$ we get,

$$0 \leq \overline{\mathrm{UB}} - \overline{\mathrm{LB}} \leq \Phi(\bar{x}, \bar{y}) - \Phi_L(\bar{x}, \bar{y}, \bar{w}) \leq 0 \tag{3.7}$$

which means that equality must hold throughout, and so, $\overline{\mathrm{UB}} = \overline{\mathrm{LB}}$. Since $\mathrm{LB}_k \leq z^* \leq \mathrm{UB}_k \ \forall k \in K$, we deduce that $\overline{\mathrm{LB}} = z^* = \overline{\mathrm{UB}}$, and this completes the proof. □

COROLLARY 2. *Along any infinite branch of the branch-and-bound tree, any accumulation point of the sequence of linear programming solution iterates generated at the nodes solves BLP*(Ω).

Proof. By selecting the index sets $K \supseteq K_1 \supseteq K_2$ as subsets of the index set corresponding to the given convergent subsequence, and proceeding as in the proof of Theorem 3, we have by (3.7) that $\Phi(\bar{x}, \bar{y}) = \Phi_L(\bar{x}, \bar{y}, \bar{w})$. But

$$z^* \geq \overline{\mathrm{LB}} = \Phi_L(\bar{x}, \bar{y}, \bar{w}) \text{ since } \mathrm{LB}_k = \Phi_L(x^{kt(k)}, y^{kt(k)} w^{kt(k)}) \text{ for all } k \in K,$$

and so we must have $z^* = \Phi(\bar{x}, \bar{y})$. This completes the proof. □

REMARK 3. Note from the convergence proof, that one need not necessarily branch by splitting both x and y-variable intervals within the algorithm. Branching could be made on the set of x or y-variable intervals alone, according to which of n or m is smaller, or according to which of x or y has a fewer number of components appearing in the cross-product terms in the objective function. In their branch-and-bound cutting plane algorithm for globally minimizing a function $f(x, y)$ over a certain closed set, Muu and Oettli [19] branch only on the y-space using the iterates obtained from lower bounding subprograms. Although the admissibility of such a rule in the context of the foregoing type of algorithm (including Al-Khayyal and Falk's [2] algorithm) was heretofore unknown, its consequence is fairly intuitive, since if for example $n = 1$ or 2, we can expect problem difficulty to be relatively low, even if m is fairly high. Hence, we have the following rule.

Branching Rule 3. Suppose that we have chosen the x-variables for splitting intervals (the case of the y-variables is treated similarly). Find (p, q) as in (3.3a), and partition

$$l_p \leq x_p \leq u_p \text{ into } l_p \leq x_p \leq \bar{x}_p \text{ and } \bar{x}_p \leq x_p \leq u_p$$

COROLLARY 3. *Theorem 3 continues to hold under Branching Rule 3.*

Proof. Note that (3.6) continues to hold, and in the limit, $\bar{x}_p = \bar{l}_p$ or $\bar{x}_p = \bar{u}_p$ also holds. Hence, (3.7) holds true, and the required result follows. □

REMARK 4. Suppose that we are using Branching Rule 3, and that the original bilinear program has only equality constraints. Then in constructing RLT products we can multiply the equality constraints with only the x-variables (to avoid

creation of $y_i y_j$ products since we are splitting the x-intervals), and also use products of the bound factors $(x_i - l_i) \geq 0$ and $(u_i - x_i) \geq 0$ on each other, and on the other y-variable bound factors $(y_i - L_j) \geq 0$ and $(U_j - y_j) \geq 0$, without again using products of the y-factors on each other. Since the original problem BLP(Ω) does not contain $y_i y_j$ cross-products, this might give a manageable and tight enough relaxation.

ENHANCEMENT OF ALGORITHMIC FATHOMING POWER

In any branch-and-bound algorithm, an active search for good quality incumbent solutions can greatly improve the computational efficiency of the algorithm by enhancing its fathoming power, without hampering any convergence arguments. Hence, instead of letting $\mathrm{UB}_{kt} = \Phi(x^{kt}, y^{kt})$ at node Φ^{kt}, we could let UB_{kt} equal some improved heuristic solution value found, and Theorem 3 would continue to hold. Such a heuristic may be implemented as follows.

Suppose that at some node, we have solved a linear program LP(Ω) and obtained a solution $(\bar{x}, \bar{y})$. As in Al-Khayyal and Falk [2], we can minimize a first order approximation of $\Phi(\cdot, \cdot)$ at $(\bar{x}, \bar{y})$, but *bound the variation about* $(\bar{x}, \bar{y})$. Hence, we solve the linear program

$$\bar{\mathrm{L}} = \text{Minimize } \{c^t x + d^t y + x^t(G\bar{y}) + (\bar{x}^t G)y \colon (x, y) \in Z \cap \Omega \cap \bar{\Omega}_\delta\} \tag{3.8a}$$

where

$$\bar{\Omega}_\delta = \{(x, y) \colon |x_i - \bar{x}_i| \leq (u_i - l_i)\delta \,\forall i\,, \quad \text{and } |y_j - \bar{y}_j| \leq (U_j - L_j)\delta \;\forall j\}$$
$$\text{for some } 0 < \delta \leq 1\,. \tag{3.8b}$$

This is a (trust region) modified version of the *Acceleration of Convergence Technique* (*ACT*) used in Al-Khayyal and Falk [2]. Similar to ACT, the next step searches along the direction $(\hat{x}, \hat{y}) - (\bar{x}, \bar{y})$, where $(\hat{x}, \hat{y})$ solves (3.8), in order to possibly find an improved incumbent. In our computational implementation of the algorithm, we used this heuristic whenever a *new* incumbent was found.

4. Computational Experience

In this section, we report on our computational experience using available test problems from the literature, as well as a selected set of pseudo-randomly generated problems with coefficients drawn uniformly from the interval $[-100, 100]$. For the sake of comparison we also implemented a relaxed variant of our algorithm in which the linear program LP'(Ω) defined by Theorem 1 is used to compute lower bounds. As shown in Theorem 1, this version is tantamount to computing lower bounds via convex envelopes of $g_{ij}x_i y_j$ over the rectangles $l_i \leq x_i \leq u_i$, $L_j \leq y_j \leq U_j$ $\forall(i, j)$, and is therefore equivalent to Al-Khayyal and

Falk's [2] algorithm for the case $G = I$, and is a direct generalization of their algorithm for the case $G \neq I$. We call this version the *Convex Envelope over Hyper-rectangles* (*CEH*)-based algorithm. For our lower bounding linear program, we first solve LP'(Ω), and if this does not fathom the node subproblem, we generate and solve LP(Ω) as defined in Section 2, except for the constraints in (2.2g)–(2.2j) for $i \neq j$. The omission of these constraints reduces the size of LP(Ω) without deteriorating its objective value for most instances. Both algorithms have been coded in Standard Fortran 77 and have been implemented in double precision arithmetic on an IBM 3090 computer. For solving the linear programming lower bounding problems, we have used the commercial package MINOS 5.1 developed by Murtagh and Saunders at the Stanford Optimization Laboratory, Stanford University, with an optimality accuracy tolerance of 1.0×10^{-6}. The ε-termination tolerance used for our algorithm is given by Remark 2 with $\varepsilon_1 = 0.001$ and $\varepsilon_2 = 0.01$. The same ε-tolerance was used for the CEH-based algorithm.

Table I gives the results on test problems from the literature, where the abbreviations are the author's initials as noted in the References section. Observe that our algorithm solved all the problems except AF0 at node zero itself via the initial linear program LP(Ω). Problem AF0 happens to be a jointly constrained bilinear program [*BLP*(*JC*)] with a nonextremal boundary point optimum. From our experience, such problems appear to pose the greatest difficulty. The CEH-based algorithm was unable to solve this problem within the preset limit of 103 nodes reached in 55 CPU secs. Also, problems AL1, AL2, ZW1, and AF3 are all jointly constrained problems, but ones that exhibit extreme point optimal solutions. Such BLP(JC) problems appear to be relatively less hard to solve. Note again that AF3 remains unsolved within the limit of 103 nodes reached after 115 CPU secs using the CEH-based algorithm. The remaining problems in Table I are all separably constrained problems [*BLP*(*SC*)], and this class of problems seems to pose the least level of difficulty. Both algorithms solve these problems with comparable efficiency. Of noteworthy mention is Problem SS1 which was solved by Sherali and Shetty [24] after generating two polar cuts over two major loop iterations. This was solved at node zero by our algorithm in 14.5 CPU secs, and was solved by the CEH-based algorithm after exploring 21 nodes in 17 CPU secs.

We remark here that Al-Khayyal and Falk [2] suggest a heuristic branching scheme that partitions a node into (two or) four subnodes based on the solution produced after applying the Acceleration of Convergence Technique (ACT), mentioned in Section 3 above. This significantly improves the computational performance, where problems AF0, AF1, AF2, and AF3 are respectively solved after enumerating 9, 5, 1, and 13 nodes. In our computations, we have used ACT only to obtain improved upper bounds, while preserving the theoretically admissible branching scheme.

Since there appears to be a distinction between the solution difficulty of problems from the classes BLP(SC) and BLP(JC), we pseudo-randomly gener-

Table I. Jointly and separably constrained BLP's from the literature

Algorithm	Source	BLP	LP	LB_0	UB_0	UB_f	N	N_{opt}	N_{amax}	CPU
	AL1	[2, 1]	[5, 16]	−2.500	−2.500	−2.500	0	0	1	4
	AL2	[2, 1]	[5, 16]	2.000	2.000	2.000	0	0	1	4.5
	KK1	[2, 4]	[14, 74]	3.000	3.000	3.000	0	0	1	3.6
Proposed	WZ1	[2, 4]	[5, 40]	−21.866	−21.866	−21.866	0	0	1	4
Algorithm	SS1	[4, 8]	[14, 136]	9.000	9.000	9.000	0	0	1	14.5
	AF0	[2, 2]	[5, 23]	−1.500	−0.9375	−1.08333	11	10	7	14
	AF1	[10, 10]	[65, 395]	−45.3797	−45.3797	−45.3797	0	0	1	12
	AF2	[10, 10]	[65, 395]	−42.9625	−42.9625	−42.9625	0	0	1	15
	AF3	[10, 13]	[65, 495]	−794.855	−794.855	−794.855	0	0	1	29
Algorithm	Source	BLP	LP	LB_0	UB_0	UB_f	N	N_{opt}	N_{amax}	CPU
	AL1	[2, 1]	[5, 5]	−2.800	−0.800	−2.500	11	9	3	5
	AL2	[2, 1]	[5, 5]	2.000	2.000	2.000	0	0	1	3.5
	KK1	[2, 4]	[14, 20]	3.000	3.000	3.000	0	0	1	3
CEH-	ZW1	[2, 4]	[5, 8]	−21.866	−21.866	−21.866	0	0	1	3.5
Based	SS1	[4, 8]	[14, 24]	−2074.00	38.00	9.000	21	8	20	17
Algorithm	AF0	[2, 2]	[5, 6]	−3.000	−0.0750	−1.08333	103+	51	20	55+
	AF1	[10, 10]	[65, 110]	−46.005	−45.3797	−45.3797	11	2	11	14.5
	AF2	[10, 10]	[65, 110]	−43.125	−42.475	−42.9625	7	4	3	10
	AF3	[10, 13]	[65, 113]	−945.451	3210.82	−794.855	103+	35	34	115+

Legend: BLP = Size of original BLP[$n + m$, # constraints]
LP = Size of LP(NODE) relaxation [# variables, # constraints]
UB_0 = Initial Upper Bound
UB_f = Final Incumbent
N = Number of nodes explored
N_{opt} = Node at which optimum is found
N_{amax} = Maximum number of active nodes per stage
CPU = Total execution time (CPU seconds)
+ = Exceeded preset limit

ated problems from both classes to further test our algorithm. Table II provides results on a sample of BLP(SC) problems. Again, all problems in this class were solved at node zero itself by our lower bounding LP(Ω) problem. Of course, one may not expect this to be generally true, since otherwise, the NP-Hard quadratic concave minimization problems would be polynomially solvable. The CEH-based algorithm, on the other hand, experienced wider gaps between the initial lower bound and the optimal value, and needed to explore several more nodes with substantially greater effort, being unable to solve several problem instances within the preset node limit.

Table III gives results on BLP(JC) problems. The problems where both algorithms generated zero (or very few) nodes are evidently easier to solve. Of the others, the CEH-based algorithm required far greater effort, with our algorithm requiring an enumeration beyond node zero for only two instances.

Finally, we remark on a particular, noteworthy, computational experiment.

REMARK 5. *4-Node vs. 2-Node Partitioning Strategy*. Recall that Al-Khayyal and Falk's [2] algorithm is based on a 4-node branching scheme, while we have shown that a 2-node partitioning process is also theoretically admissible. We attempted the 4-node partitioning scheme on the test problems from the literature, and found a substantial deterioration in both algorithms. For Problem AF0 where our algorithm requires branching in Table I, the 4-node scheme enumerated 93 nodes in 110 CPU secs. For the CEH-based algorithm, this scheme resulted in the following values of (N, CPU) for AL1, SS1, AF0, AF1, AF2, and AF3, respectively: (81, 43), (201+, 125.5+), (201+, 115+), (41, 42.5), (37, 45) and (201+, 226+). Evidently, the flexibility of partitioning on either x or y-variables and at different solutions as afforded by the 2-node branching process, pays off very favorably.

5. Convex Hull Characterizing Properties of RLT for Special Instances

In this section, we provide some motivation for the use of RLT and its observed computational success by demonstrating that a particular application of this technique to certain triangular and quadrilateral D-polytopes in R^2 (as introduced by Sherali and Alameddine [23]), followed by a projection onto the original variable space, produces the exact convex hull representation for these cases. Hence, the reformulation technique constructs the *exact* convex envelope of the function xy over such polytopes, and so the bounding linear program recovers the optimal objective value of the underlying bilinear program. This property plays two roles. First, it motivates the use of RLT in producing sharp representations for BLP. Second, if partitioning is performed in Al-Khayyal and Falk's [2] algorithm so that Ω is decomposed into the cartesian product of such polytopes, then the convex envelope based bound would match that produced by such a restricted application of RLT. This statement is in the same spirit as Theorem 1.

Table II. Results on pseudo-randomly generated BLP(SC)'s

Algorithm	Prob. Name	BLP	LP	LB_0	UB_0	N	N_{opt}	N_{amax}	CPU
	SC1	[2:4, 4]	[27, 112]	−19558.0	−19558.0	0	0	1	4.1
	SC2	[2:4, 8]	[27, 190]	−209300.0	−209300.0	0	0	1	4.4
	SC3	[3:5, 10]	[44, 309]	−19.0073	−19.0073	0	0	1	5.4
	SC4	[3:5, 12]	[44, 366]	0.000	0.000	0	0	1	3.7
Proposed	SC5	[6:3, 12]	[54, 405]	−136.249	−136.249	0	0	1	7.8
Algorithm	SC6	[7:4, 12]	[77, 499]	−206462.66	−206462.66	0	0	1	14.1
	SC7	[7:4, 12]	[77, 430]	−298276.11	−298276.11	0	0	1	20
	SC8	[5:4, 12]	[54, 413]	−292047.78	−292047.69	0	0	1	11
	SC9	[4:5, 12]	[54, 413]	−1472814.27	−1472814.29	0	0	1	14.3
	SC10	[7:6, 10]	[104, 532]	−1160018.44	−1160018.44	0	0	1	27
	SC11	[7:7, 12]	[119, 664]	−1923088.82	−1923088.82	0	0	1	69
Algorithm	Prob. Name	BLP	LP	LB_0	UB_0	N	N_{opt}	N_{amax}	CPU
	SC1	[2:4, 4]	[27, 36]	−19937.0	−13238.0	103+	80	3	89+
	SC2	[2:4, 8]	[27, 40]	−209303.0	−208907.0	3	1	2	4.5
	SC3	[3:5, 10]	[44, 70]	−47046.36	6525.57	103+	64	44	105+
	SC4	[3:5, 12]	[44, 72]	0.000	0.000	0	0	1	3.7
CEH-	SC5	[6:3, 12]	[54, 84]	−1956.23	−55.59	103+	88+	7	81
Based	SC6	[7:4, 12]	[77, 124]	−243492.84	−140628.17	25	14	4	30
Algorithm	SC7	[7:4, 12]	[77, 122]	−336896.37	29848.22	21	13	3	25
	SC8	[5:4, 12]	[54, 92]	−1142473.77	14186.38	103+	–	29	84+
	SC9	[4:5, 12]	[54, 92]	−1523769.34	−1468686.77	15	14	3	14
	SC10	[7:6, 10]	[104, 178]	−1516842.22	−5727.09	17	15	4	28
	SC11	[7:7, 12]	[119, 208]	−2349670.31	−1610111.10	103+	91	37	163+

Legend: BLP = Size of original BLP[$n + m$, # constraints]
LP = Size of LP(NODE) relaxation [# variables, # constraints]
UB_0 = Initial Upper Bound
UB_f = Final Incumbent
N = Number of nodes explored
N_{opt} = Node at which optimum is found
N_{amax} = Maximum number of active nodes per stage
CPU = Total execution time (CPU seconds)
+ = Exceeded preset limit

Table III. Results on pseudo-randomly generated BLP(JC)'s

Prob. Name	BLP	Proposed Algorithm LP	N	N_{opt}	CPU	CEH-based Algorithm LP	N	N_{opt}	CPU
JC1	[4, 7]	[14, 119]	0	0	2.5	[14, 23]	0	0	1.5
JC2	[4, 7]	[14, 119]	19	17	32	[14, 23]	39	37	33
JC3	[4, 7]	[14, 119]	0	0	4	[14, 23]	17	17	15
JC4	[6, 7]	[27, 173]	0	0	4.2	[27, 43]	63+	–	80+
JC5	[8, 7]	[44, 235]	0	0	14	[44, 71]	3	3	6.5
JC6	[8, 7]	[44, 235]	0	0	19	[44, 71]	63+	–	120+
JC7	[8, 3]	[44, 145]	5	4	7	[44, 67]	103+	–	120+
JC8	[10, 6]	[65, 277]	0	0	5	[65, 107]	0	0	7
JC9	[10, 7]	[65, 305]	0	0	7	[65, 107]	0	0	6
JC10	[10, 7]	[65, 305]	0	0	6	[65, 107]	27	26	39
JC11	[12, 7]	[90, 383]	0	0	13	[90, 151]	0	0	8
JC12	[12, 7]	[90, 383]	0	0	5	[90, 151]	0	0	5
JC13	[14, 7]	[119, 469]	0	0	6.3	[119, 203]	0	0	5

Legend: BLP = Size of original BLP[$n + m$, # constraints]
LP = Size of LP(NODE) relaxation [# variables, # constraints]
N = Number of nodes explored
N_{opt} = Node at which optimum is found
CPU = Total execution time (CPU seconds)
+ = Exceeded preset limit

As a preliminary, we remark that the construction of the convex envelope of a function over a given set and the construction of the closure of the convex hull (denoted cl.conv) of its epigraph over this set, are identical processes. This is highlighted below.

THEOREM 4. *Let $X \subseteq R^n$ be a nonempty closed, convex set, and let $f: X \to R$ be a continuous function. Define the epigraph of f over X as*

$$S = \{(x, z): z \geq f(x), x \in X\},$$

Let f_X denote the convex envelope of f over X, and consider the epigraph

$$S' = \{(x, z): z \geq f_X(x), x \in X\}$$

of f_X over X. Then cl.conv$(S) = S'$.

Proof. (Horst and Tuy [12, Lemma IV.1] show that $S' = \text{conv}(S)$ when X is compact. We sketch a brief extension for the case of X being unbounded.). Now, if $(\bar{x}, \bar{z}) \in S$, then $\bar{x} \in X$ and $\bar{z} \geq f(\bar{x}) \geq f_X(\bar{x})$, and so $(\bar{x}, \bar{z}) \in S'$. Hence, $S \subseteq S'$ and since S' is closed and convex, we have cl.conv$(S) \subseteq S'$. Conversely, suppose that $(\bar{x}, \bar{z}) \in S'$. From Falk [9], we know that

$$f_X(\bar{x}) = \inf\{z: (\bar{x}, z) \in \text{conv}(S)\} \tag{5.1}$$

and so $[\bar{x}, f_X(\bar{x})] \in$ cl.conv(S). Since $\bar{z} \geq f_X(\bar{x})$, it follows that $(\bar{x}, \bar{z}) \in$ cl.conv(S), and this completes the proof. □

REMARK 6. Note that the above result shows that the epigraph of the convex envelope of f over X defines the closure convex hull of the epigraph of f over X. Note also that conv(S) need not necessarily be closed, as for example when $f(x) = 1 - e^{-x}$ and $X = \{x: x \geq 0\}$. Also, of related interest, if we replace $z \geq f(x)$ by $z = f(x)$ in S, then S' inherits an additional constraint $z \leq f^X(x)$, where f^X is the concave envelope of f over X, and then we can again show that cl.conv$(S) = S'$.

Below, we give two theorems which make use of Theorem 4 to show that a particular application of RLT produces the closure of the convex hull of the set $\{(x, y, z): z \geq xy, (x, y) \in X\}$, when X is a triangular or quadrilateral D-polytope in R^2. As in Sherali and Alameddine [23], a D-polytope is one with no finite, positively sloped edge. Over such polytopes, Sherali and Alameddine provide an explicit convex envelope characterization for bilinear functions, showing that the epigraph of the envelope is polyhedral. The results below demonstrate that a suitable RLT process recovers this characterization for the stated cases.

Henceforth, notationally, RLT(2) shall denote an application of the RLT in which the original defining constraints of the polyhedron are multiplied by each other pairwise as in (2.2). Also, let us define Z_2 and Z_{L2} to be respectively regions defined by the new nonlinear constraints generated by applying RLT(2), and by their linearized counterpart following the substitution of type (2.3).

TRIANGULAR D-POLYTOPES

To begin, consider the triangular D-polytope Z shown in Figure 1. The system of defining constraints associated with this polytype are

$$-\left[\frac{1}{r}\right]x - \left[\frac{1}{s}\right]y + 1 \geq 0 \tag{5.2a}$$

$$\left[\frac{s-q}{sp}\right]x + \left[\frac{1}{s}\right]y - 1 \geq 0 \tag{5.2b}$$

$$\left[\frac{1}{r}\right]x + \left[\frac{r-p}{rq}\right]y - 1 \geq 0\,. \tag{5.2c}$$

Note that there is no loss of generality in conveniently defining the origin relative to the polytope as shown. That is, given a D-polytope, if the origin is translated and then RLT(2) is applied to the resulting defining inequalities, the representation obtained can be verified to be equivalent to that obtained by applying RLT(2) to the original defining inequalities themselves. (See [1] for details.)

For this triangular polytope, we have the following result which states that RLT(2), with projection, constructs the convex envelope of the bilinear function $f(x, y) = xy$ over the region Z, and hence produces cl.conv $\{(x, y, z)\colon z \geq xy, (x, y) \in Z\}$.

THEOREM 5. Consider the triangular D-polytope $Z \subset R^2$ shown in Figure 1. Using RLT(2), generate the 3 constraints of Z_{L2} obtained by linearizing the constraint products $\{(5.2a)\cdot(5.2b), (5.2a)\cdot(5.2c)$, and $(5.2b)\cdot(5.2c)\}$ using the substitution $w = xy$, $X = x^2$, and $Y = y^2$. Define

$$S = \{(x, y, z)\colon z \geq xy, (x, y) \in Z\}\,,$$

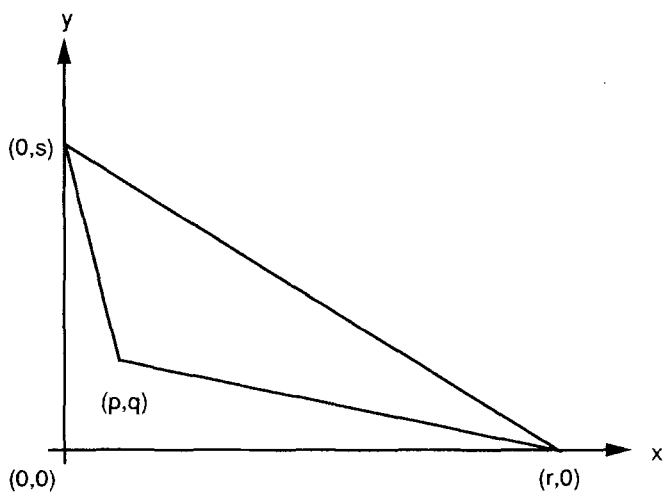

Fig. 1. A triangular D-polytope in R^2.

and denote

$$Z^p = \{(x, y, z): z \geq w, (x, y) \in Z, (x, y, w, X, Y) \in Z_{L2}\}$$

to be the projection onto the (x, y, z)-space of the set obtained by applying RLT(2). Then $Z^p = \text{cl.conv(S)}$.

Proof. Given any $(x, y, z) \in S$, by defining $w = xy$, $X = x^2$, and $Y = y^2$, it follows by construction that $(x, y, z) \in Z^p$. Hence, $S \subseteq Z^p$, and since Z^p is polyhedral, this means that $\text{cl.conv}(S) \subseteq Z^p$. To complete the proof, we need to show that $Z^p \subseteq \text{cl.conv}(S)$. By Theorem 4, denoting $f(x, y) = xy$, we have that

$$\text{cl.conv}(S) = \{(x, y, z): z \geq f_Z(x, y), (x, y) \in Z\} .$$

Hence, given any $(x, y, z) \in Z^p$, we need to show that $z \geq f_Z(x, y)$. Noting the constraints of Z^p, it is sufficient to show that the constraints of Z_{L2} imply that $w \geq f_Z(x, y)$. This is the task undertaken below in order to prove the theorem.

First of all, note that from Sherali and Alameddine [23], it is readily verified that

$$f_Z(x, y) = \left[\frac{-(pqs)x - (pqr)y + rspq}{(rs - sp - rq)} \right]. \tag{5.3}$$

Now, the constraints of Z_{L2} are given as follows.

$$-\left[\frac{s-q}{rsp}\right]X - \left[\frac{1}{s^2}\right]Y \geq -\left[\frac{rs+sp-rq}{rsp}\right]x - \left[\frac{2}{s}\right]y + \left[\frac{rs+sp-rq}{rs^2p}\right]w + 1 \tag{5.4a}$$

$$-\left[\frac{1}{r^2}\right]X + \left[\frac{p-r}{rsq}\right]Y \geq -\left[\frac{2}{r}\right]x - \left[\frac{rs+rq-sp}{rsq}\right]y + \left[\frac{rs+rq-sp}{rs^2q}\right]w + 1 \tag{5.4b}$$

$$\left[\frac{s-q}{rsp}\right]X + \left[\frac{r-p}{rsq}\right]Y \geq \left[\frac{rs+sp-rq}{rsp}\right]x + \left[\frac{rs+rq-sp}{rsq}\right]y - \left[\frac{rs-sp-rq+2pq}{rspq}\right]w - 1 . \tag{5.4c}$$

Denote the above set of inequalities (5.4) as $P\alpha \geq Q\beta + b$, where $\alpha = (X, Y)^t$, and $\beta = (x, y, w)^t$. Then, by linear programming duality, there exists an α that satisfies $P\alpha \geq Q\beta + b$ for a given β if and only if

$$(Q\beta + b)^t\pi \leq 0 \quad \text{for any } \pi \in \Lambda = \{\pi: P^t\pi = 0, e^t\pi = 1, \pi \geq 0\} . \tag{5.5}$$

The set Λ is explicitly given below.

$$-\left[\frac{s-q}{rsp}\right]\pi_1 - \left[\frac{1}{r^2}\right]\pi_2 + \left[\frac{s-q}{rsp}\right]\pi_3 = 0 \tag{5.6a}$$

$$-\left[\frac{1}{s^2}\right]\pi_1 - \left[\frac{r-p}{rsq}\right]\pi_2 + \left[\frac{r-p}{rsq}\right]\pi_3 = 0 \tag{5.6b}$$

$$\pi_1 + \pi_2 + \pi_3 = 1 \tag{5.6c}$$

$$\pi_1,\ \pi_2,\ \pi_3 \geqslant 0\,. \tag{5.6d}$$

Note that by (5.5),

$$\beta \equiv (x,\ y,\ w) \in Z_{L2} \text{ if and only if } (\bar{\pi})^t(Q\beta + b) \leqslant 0 \quad \forall \bar{\pi} \in \text{vert}(\Lambda)\,, \tag{5.7}$$

where vert(Λ) denotes the vertices of Λ. Furthermore, note from (5.4) that

$$(Q\beta + b)^t\bar{\pi} = c_1 x + c_2 y + c_{12} w + (\bar{\pi}_1 + \bar{\pi}_2 - \bar{\pi}_3) \tag{5.8a}$$

where c_1, c_2, and c_{12} are given as follows:

$$c_1 = -\left[\frac{rs + sp - rq}{rsp}\right]\bar{\pi}_1 - \left[\frac{2}{r}\right]\bar{\pi}_2 + \left[\frac{rs + sp - rq}{rsp}\right]\bar{\pi}_3 \tag{5.8b}$$

$$c_2 = -\left[\frac{2}{s}\right]\bar{\pi}_1 - \left[\frac{rs + rq - sp}{rsq}\right]\bar{\pi}_2 + \left[\frac{rs + rq - sp}{rsq}\right]\bar{\pi}_3 \tag{5.8c}$$

$$c_{12} = \left[\frac{rs + sp - rq}{r^2 p}\right]\bar{\pi}_1 + \left[\frac{rs + rq - sp}{r^2 sq}\right]\bar{\pi}_2 + \left[\frac{rs - sp - rq + 2pq}{rspq}\right]\bar{\pi}_3\,. \tag{5.8d}$$

Now, it can be readily verified that Λ has a single vertex $\bar{\pi}$ given as the solution to the system (5.6a)–(5.6c). The solution to this subsystem is given below in closed-form

$$\bar{\pi}_1 = \frac{s(r-p)}{(3rs - sp - rq)} \geqslant 0\,, \quad \bar{\pi}_2 = \frac{r(s-q)}{(3rs - sp - rq)} \geqslant 0\,,$$

$$\bar{\pi}_3 = \frac{rs}{(3rs - sp - rq)} \geqslant 0 \tag{5.9}$$

where the nonnegativity follows by noting that rs is an upper bound on the values of sp and rq. Letting $M = 3rs - sp - rq$ be a scaling parameter, and using (5.9) in (5.8), we have

$$Mc_1 = \left(\frac{-rs + sp + rq}{r}\right) \tag{5.10a}$$

$$Mc_2 = \left(\frac{-rs + sp + rq}{s}\right) \tag{5.10b}$$

$$Mc_{12} = -\left[\frac{(rs - sp - rq)^2}{rspq}\right]. \tag{5.10c}$$

Furthermore, from (5.9), we have,

$$M(\bar{\pi}_1 + \bar{\pi}_2 - \bar{\pi}_3) = rs - sp - rq\,. \tag{5.11}$$

Hence, putting (5.7) and (5.8a) together, we have that

$$(x, y, w) \in Z_{L2} \text{ if and only if } M[c_1x + c_2y + c_{12}w + (\bar{\pi}_1 + \bar{\pi}_2 - \bar{\pi}_3)] \leq 0 .$$

Noting that $c_{12} < 0$, and writing the foregoing condition as

$$w \geq -\left(\frac{c_1}{c_{12}}\right)x - \left(\frac{c_2}{c_{12}}\right)y - \left[\frac{(\bar{\pi}_1 + \bar{\pi}_2 - \bar{\pi}_3)}{c_{12}}\right],$$

we see from (5.3), (5.10), and (5.11) that this is precisely $w \geq f_Z(x, y)$. This completes the proof. □

QUADRILATERAL D-POLYTOPES

Next, consider the quadrilateral D-polytope Z shown in Figure 2. The system of defining constraints associated with this polytope Z is

$$-\left[\frac{1}{r}\right]x - \left[\frac{1}{s}\right]y + 1 \geq 0 \tag{5.12a}$$

$$\left[\frac{s-q}{sp}\right]x + \left[\frac{1}{s}\right]y - 1 \geq 0 \tag{5.12b}$$

$$\left[\frac{q-u}{tq-pu}\right]x + \left[\frac{t-p}{tq-pu}\right]y - 1 \geq 0 \tag{5.12c}$$

$$\left[\frac{1}{r}\right]x + \left[\frac{r-t}{ru}\right]y - 1 \geq 0 . \tag{5.12d}$$

Similar to Theorem 5, for the quadrilateral polytope Z shown in Figure 2, we

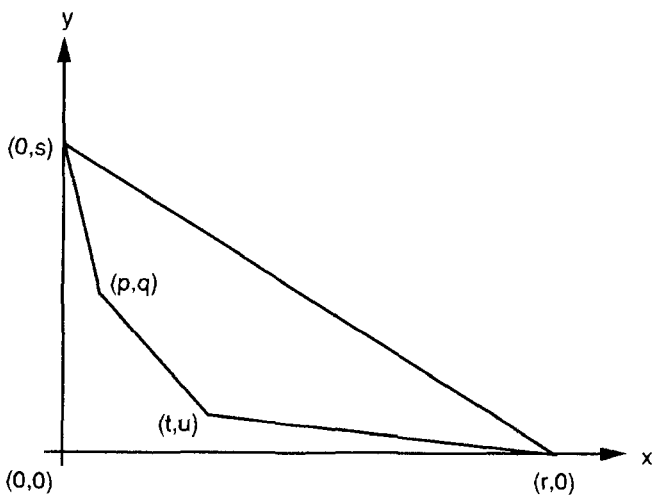

Fig. 2. A quadrilateral D-polytope in R^2.

have the following result. (This result also holds for quadrilateral D-polytopes having other shapes than that in Figure 2, including rectangles.)

THEOREM 6. *Consider the quadrilateral D-polytope* $Z \subset R^2$ *shown in Figure 2. Using RLT(2), generate the 6 constraints of* Z_{L2} *by linearizing the constraint products* $\{(3.4a)\cdot(3.4b),\ (3.4a)\cdot(3.4c),\ (3.4a)\cdot(3.4d),\ (3.4b)\cdot(3.4c),\ (3.4b)\cdot(3.4d)$ *and* $(3.4c)\cdot(3.4d)\}$, *by using the substitution* $w = xy$, $X = x^2$, $Y = y^2$. *Define* $S = \{(x, y, z):\ z \geq xy,\ (x, y) \in Z\}$, *and denote* $Z^p = \{(x, y, z):\ z \geq w,\ (x, y) \in Z,\ (x, y, w, X, Y) \in Z_{L2}\}$ *to be the projection onto the* (x, y, z)*-space of the set obtained by applying RLT*(2). *Then* $Z^p = cl.conv(S)$.

Proof. Following the proof of Theorem 5, we identically have that $\text{cl.conv}(S) \subseteq Z^p$, and in order to show that $Z^p \subseteq \text{cl.conv}(S)$, we need to show that the constraints of Z_{L2} imply that $w \geq f_Z(x, y)$.

Toward this end, let $P\alpha \geq Q\beta + b$ denote the inequalities of Z_{L2} generated by the pairwise products given in the theorem, *in that order*, where $\alpha = (X, Y)^t$ and $\beta = (x, y, w)^t$. Then the projection Z^p of this constraint set onto the space of the variables $\beta = (x, y, w)^t$ is given by

$$Z^p \equiv \{(x, y, w):\ (x, y) \in Z,\ (x, y, w)Q^t\pi^k + b^t\pi^k \leq 0,\quad \forall k = 1, \ldots, K\},$$

where π^k, $k = 1, \ldots, K$, are the vertices of the set $\Lambda = \{\pi:\ P^t\pi = 0,\ e^t\pi = 1,\ \pi \geq 0\}$, and where $e^t = (1, \ldots, 1)$. Note that the vector π equals $(\pi_1, \ldots, \pi_6)^t$, and that Λ has three equality constraints that can be verified to be linearly independent. Hence, extreme points of Λ have three basic variables. Now, consider two cases.

CASE 1. Suppose that the ratio

$$\left[\frac{tu(sp + rq - rs)}{pq(st + ru - rs)}\right] > 1. \tag{5.13}$$

Consider the following two choices of bases for Λ, where the three designated nonzero components of each solution π^q, $q = 1, 2$, are considered as basic.

$$\pi^1 = (\bar{\pi}_1, \bar{\pi}_2, 0, 0, \bar{\pi}_5, 0)^t \tag{5.14a}$$

$$\pi^2 = (0, \bar{\pi}_2, 0, 0, \bar{\pi}_5, \bar{\pi}_6)^t. \tag{5.14b}$$

It is shown in Alameddine [1] Appendix A, through tedious algebraic manipulations, that these solutions are indeed vertices of Λ, and that the two constraints $(x, y, w)Q^t\pi^k + b^t\pi^k \leq 0$, $k = 1, 2$, define the convex envelope of $f(x, y) = xy$ over Z. In fact, these constraints represent the convex envelopes of f over the triangular regions ABD and BCD, respectively, shown in Figure 3(a), which are described by the following explicitly stated supports:

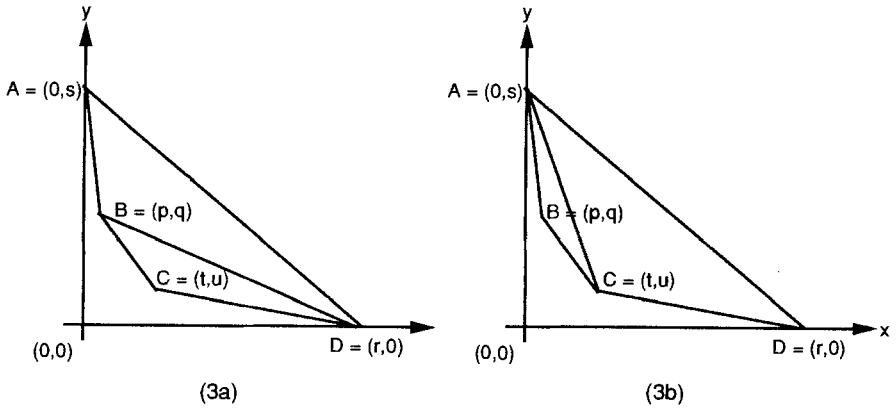

Fig. 3. Triangular decomposition for Theorem 6.

$$w \geq \left[\frac{-(spq)x - (rpq)y + rspq}{(rs - rq - sp)} \right] \tag{5.15a}$$

$$w \geq \left[\frac{u(t - \beta_4^*)}{(t - r)} \right] x + \left[\frac{pq(t - r) + tu(r - p)}{q(t - r) + u(r - p)} \right] y - \left[\frac{ru(t - \beta_4^*)}{(t - r)} \right],$$

where

$$\beta_4^* = \left[\frac{pq(t - r) + tu(r - p)}{q(t - r) + u(r - p)} \right].$$

Hence, in Case 1, the constraints of Z_{L2} imply that $w \geq f_Z(x, y)$.

CASE 2. On the other hand, suppose that the ratio in (5.13) satisfies

$$0 \leq \left[\frac{tu(sp + rq - rs)}{pq(st + ru - rs)} \right] \leq 1. \tag{5.16}$$

Consider the following two choices of bases for Λ, where the three designated nonzero components of each solution π^q, $q = 1, 2$ are considered as basic.

$$\pi^3 = (0, \bar{\pi}_2, \bar{\pi}_3, 0, \bar{\pi}_5, 0)^t \tag{5.17a}$$

$$\pi^4 = (0, \bar{\pi}_2, 0, \bar{\pi}_4, \bar{\pi}_5, 0)^t. \tag{5.17b}$$

Again, it can be shown that these are indeed vertices of Λ, and that the two constraints $(x, y, w)Q^t\pi^k + b^t\pi^k \leq 0$, $k = 1, 2$, define the convex envelope of $f(x, y) = xy$ over Z. In fact, these constraints represent the convex envelopes of f over the triangular regions ACD and ABC, respectively, shown in Figure 3(b), which are described by the following explicitly stated supports:

$$w \geq \left[\frac{-(stu)x - (rtu)y + rstu}{(rs - ru - st)} \right] \tag{5.18a}$$

$$w \geq \left[\frac{q[t(s-q)+p(u-s)]+t(u-q)(s-q)}{t(s-q)+p(u-s)}\right]x + \left[\frac{pt(u-q)}{t(s-q)+p(u-s)}\right]y - \left[\frac{spt(u-q)}{t(s-q)+p(u-s)}\right]. \quad (5.18b)$$

Hence, we again have in Case 2, that the constraints of Z_{L2} imply that $w \geq f_Z(x, y)$, and this completes the proof.

From Figures (3a) and (3b), note that the diagonals BD and AC each partition the area of quadrilateral ABCD into two triangular regions. Theorem 6 gives the condition under which either diagonal is used in the triangular partitioning process, and states that it is dependent on the ratio:

$$\left[\frac{tu(sp+rq-rs)}{pq(st+ru-rs)}\right].$$

Geometrically, this ratio represents the relative distances of the points (p, q) and (t, u) from the longest side AD.

We remark here that the concave envelope of the function xy over Z is not available via the RLT scheme in the above cases. This follows because the concave envelope is given by the negative of the convex envelope of $-xy$ over Z. Substituting $x' = -x$, this reduces to determining the convex envelope of $x'y$ over a corresponding set Z'. However, Z' need not be a D-polytope, and in fact, the convex envelope of $x'y$ over Z' need not be polyhedral. Nonetheless, the foregoing analysis gives a theoretical motivation for using RLT. We also remark here, that for the bilinear function xy defined over a pentagonal D-polytope, RLT(2) fails to produce the convex hull representation. For example, for the

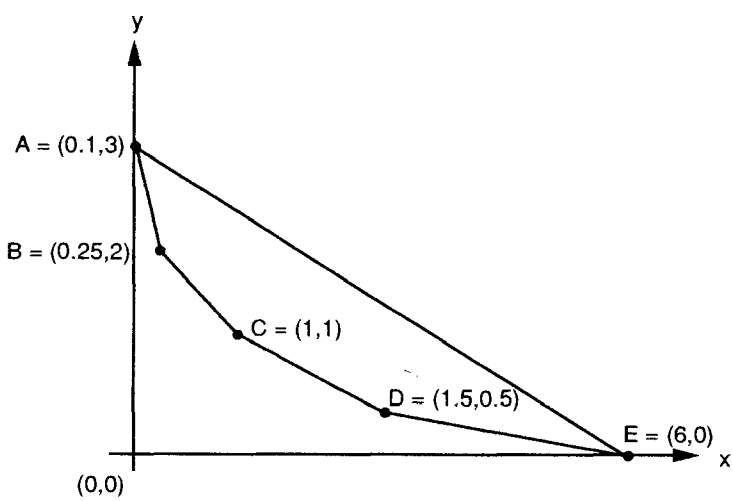

Fig. 4. Pentagon D-polytope in R^2.

pentagon shown in Figure 4, one can verify that the convex hull representation in the spirit of Theorems 5 and 6 is not produced by RLT(2), while it is produced by RLT(3), which considers products of constraints taken three at a time as well. This therefore motivates a possible consideration for higher order products.

6. Summary and Extensions

To summarize, we have presented a new Reformulation-Linearization Technique (RLT) to generate tight linear programming relaxations for (general) jointly constrained bilinear programming problems. This RLT process has been shown to yield convex hull representations for triangular and quadrilateral D-polytopes in R^2. The linear programming bounding problems, when imbedded in a branch-and-bound algorithm, yield a computationally effective algorithm that is theoretically convergent under various flexible and useful partitioning schemes. Empirically, the algorithm solves most problems at the initial node itself unless the problem has a nonextremal boundary point solution, which then necessitates the enumeration of a few nodes.

The main computational burden of the algorithm lies in the solution of the lower bounding problems, and this can be alleviated by using more efficient (e.g., interior point or Lagrangian duality-based) linear programming solvers. Alternatively, we can reduce the size of the bounding linear programs by dispensing with constraints that might not contribute significantly to the tightness of these bounds, while unduly burdening the algorithm. We highlight our preliminary attempt at such a "constraint filtering strategy" below.

REMARK 7. *Constraint Filtering Strategy.* In the first step of this strategy, we solve the CEH-based problem LP'(Ω), to obtain a solution $(\bar{x}, \bar{y}, \bar{w})$. This solution is augmented to produce $\bar{z} \equiv (\bar{x}, \bar{y}, \bar{w}, \bar{X}, \bar{Y})$ through the substitution $\bar{X}_{ij} = \bar{x}_i \bar{x}_j$, $\forall i, j$, and $\bar{Y}_{ij} = \bar{y}_i \bar{y}_j$, $\forall i, j$. The constraints of LP(Ω) are then generated in the usual manner, and for each such inequality constraint $\alpha_i z \geq \beta_i$, say, we compute the (signed) Euclidean distance $(\alpha_i \bar{z} - \beta_i)/\|\alpha^i\|$ of $\bar{z}$ from the corresponding hyperplane. If this distance exceeds a specified tolerance $\tau > 0$, the constraint is deleted. The resulting reduced problem is then used to compute a lower bound. Computationally, we found that a suitable value of τ can indeed reduce storage and effort, but such a value is highly problem dependent and therefore, an effective implementation of this strategy requires further investigation.

We are also pursuing several other avenues for future research. These include the design of a more effective RLT scheme, the use of triangular and quadrilateral D-polytope partitions of rectangles, the computational testing of transforming the bilinear term x^tGy to the form x^tz through the substitution $z = Gy$, the design of more computationally effective branching strategies as suggested by Al-Khayyal and Falk [2] with respect to their heuristic acceleration technique, and

the design of an effective constraint filtering strategy as in Remark 7. This work can also be extended to solve indefinite quadratic programs, as well as bilinearly constrained bilinear programming problems. In particular, Sherali and Tuncbilek [25] discuss an extension to solve general polynomially constrained polynomial programming problems.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. ECS-8807090 and the Air Force Office of Scientific Research under Grant No. 2304/B1. The government has certain rights in this material. We also thank an anonymous referee for detailed comments that helped improve the readability of this paper.

References

1. Alameddine, A. R. (1990), A New Reformulation-Linearization Technique for the Bilinear Programming and Related Problems with Applications to Risk Management, Ph.D. Dissertation, Department of Industrial and Systems Engineering, Blacksburg, VA 24061.
2. Al-Khayyal, F. A. and J. E. Falk (AF) (1983), Jointly Constrained Biconvex Programming, *Mathematics of Operations Research* **8**, 273–286.
3. Al-Khayyal, F. A. (1983), Jointly Constrained Bilinear Programming and Related Problems, Report, Industrial and Systems Engineering No. J-83-3, Georgia Institute of Technology, Atlanta, GA.
4. Al-Khayyal, F. A. (1986), Linear Quadratic, and Bilinear Programming Approaches to the Linear Complementarity Problem, *European Journal of Operational Research* **24**, 216–227.
5. Al-Khayyal, F. A. (1990a), Jointly Constrained Bilinear Programs: An Overview, *Journal of Computers and Mathematics with Applications* **19**(11), 53–62.
6. Al-Khayyal, F. A. (1990b), Generalized Bilinear Programming, Part I: Models, Applications and Linear Programming Relaxation, *European Journal of Operational Research*, forthcoming.
7. Al-Khayyal, F. A. and C. Larsen (AL) (1990), Global Minimization of a Quadratic Function Subject to a Bounded Mixed Integer Constraint Set, *Annals of Operations Research* **25**, 169–180.
8. Czochralska, I. (1982), Bilinear Programming, *Applictiones Mathematicae XVIII* **3**, 495–514.
9. Falk, J. E. (1969), Lagrange Multipliers and Nonconvex Programs, *SIAM Journal on Control* **7**(4), 534–545.
10. Horst, R. (1976), An Algorithm for Nonconvex Programming Problems, *Mathematical Programming* **10**, 312–321.
11. Horst, R. (1986), A General Class of Branch and Bound Methods in Global Optimization with Some New Approaches for Concave Minimization, *Journal of Optimization Theory and Applications* **51**(2), 271–291.
12. Horst, R. and H. Tuy (1990), *Global Optimization: Deterministic Approaches*, Springer-Verlag, Berlin.
13. Kalantari, B. and J. B. Rosen (1987), An Algorithm for Global Minimization of Linearly Constrained Concave Quadratic Functions, *Mathematics of Operations Research*, **12**, 544–561.
14. Konno, H. (1971) Bilinear Programming, Part I: An Algorithm for Solving Bilinear Programs, Technical Report No. 71–9, Operations Research House, Stanford University (Stanford, CA).
15. Konno, H. (1971), Bilinear Programming, Part II: Applications of Bilinear Programming, Technical Report No. 71-10, Operations Research House, Stanford University (Stanford, CA).
16. Konno, H. (1976a), A Cutting Plane Algorithm for Solving Bilinear Programs, *Mathematical Programming* **11**, 14–27.

17. Konno, H. (1976b), Maximization of a Convex Quadratic Function under Linear Constraints, *Mathematical Programming* **11**, 117–127.
18. Konno, H. and T. Kuno (KK) (1989), *Generalized Linear Multiplicative and Fractional Programming*, forthcoming.
19. Muu, L. D. and W. Oettli (1990), *Combined Branch-and-Bound and Cutting Plane Methods for Solving a Class of Nonlinear Programming Problems*, forthcoming.
20. Ritter, K. (1966) A Method for Solving Maximum Problems with a Nonconvex Quadratic Objective Function, *Wahrscheinlichkeitstheorie* **4**, 340–351.
21. Sherali, H. D. and W. P. Adams (1990a), A Hierarchy of Relaxations between the Continuous and Convex Hull Representations for Zero-One Programming Problems, *SIAM Journal on Discrete Mathematics* **3**(3), 411–430.
22. Sherali, H. D. and W. P. Adams (1990b), *A Hierarchy of Relaxations and Convex Hull Characterizations for Zero-One Mixed Integer Programming Problems*, forthcoming.
23. Sherali, H. D. and A. R. Alameddine (1990), An Explicit Characterization of the Convex Envelope of a Bivariate Bilinear Function over Special Polytopes, in Pardalos P. M. and J. B. Rosen (eds.), *Annals of OR: Computational Methods in Global Optimization* **25**, 197–210.
24. Sherali, H. D. and C. M. Shetty (SS) (1980), A Finitely Convergent Algorithm for Bilinear Progamming Problems Using Polar Cuts and Disjunctive Face Cuts, *Mathematical Programming* **19**, 14–31.
25. Sherali, H. D. and C. H. Tuncbilek (1991), A Global Optimization Algorithm for Polynomial Programming Problems Using a Reformulation-Linearization Technique, in Floudas, C. A. and P. M. Pardalos (eds.), *Recent Advances in Global Optimization*, Princeton University Press, Princeton, NJ, forthcoming. (Also, *Journal of Global Optimization* **2**, 101–112 (1992).)
26. Thieu, T. V. (1988), A Note on the Solution of Bilinear Problems by Reduction to Concave Minimization, *Mathematical Programming* **41**, 249–260.
27. Tuy, H. (1964), Concave Programming under Linear Constraints, *Soviet Math. Doklady* **5**, 1437–1440.
28. Vaish, H. and C. M. Shetty (1976), The Bilinear Programming Problem, *Naval Research Logistics Quarterly* **23**, 303–309.
29. Vaish, H. and C. M. Shetty (1977), A Cutting Plane Algorithm, for the Bilinear Programming Problem, *Naval Research Logistics Quarterly* **24**, 83–94.
30. Yajima, Y. and H. Konno (1989), Efficient Algorithms for Solving Rank Two and Rank Three Bilinear Programs, forthcoming.
31. Zwart, P. B. (ZW) (1973), Nonlinear Programming: Counterexamples to Two Global Optimization Algorithms, *Operations Research* **21**, 1260–1266.