New SOCP relaxation and branching rule for bipartite bilinear programs

Santanu S. Dey*[1], Asteroide Santana†[1], and Yang Wang‡[2]

[1]School of Industrial and Systems Engineering, Georgia Institute of Technology
[2]School of Civil and Environmental Engineering, Georgia Institute of Technology

November 27, 2018

Abstract

A bipartite bilinear program (BBP) is a quadratically constrained quadratic optimization problem where the variables can be partitioned into two sets such that fixing the variables in any one of the sets results in a linear program. We propose a new second order cone representable (SOCP) relaxation for BBP, which we show is stronger than the standard SDP relaxation intersected with the boolean quadratic polytope. We then propose a new branching rule inspired by the construction of the SOCP relaxation. We describe a new application of BBP called as the finite element model updating problem, which is a fundamental problem in structural engineering. Our computational experiments on this problem class show that the new branching rule together with an polyhedral outer approximation of the SOCP relaxation outperforms a state-of-the-art commercial global solver in obtaining dual bounds.

Keywords: Convex hull, Bilinear programming, Branching rule

1 Introduction: Bipartite bilinear program (BBP)

A quadratically constrained quadratic program (QCQP) is called as a bilinear optimization problem if every degree two term in the constraints and objective involves the product of two distinct variables. For a given instance of bilinear optimization problem, one often associates a simple graph constructed as follows: The set of vertices corresponds to the variables in the instance and there is an edge between two vertices if there is a degree two term involving the corresponding variables in the instance formulation. Strength of various convex relaxations for bilinear optimization problems can be analyzed using combinatorial properties of this graph [24, 5, 18].

*santanu.dey@isye.gatech.edu
†asteroide.santana@gatech.edu
‡yang.wang@ce.gatech.edu

When this graph is bipartite, we call the resulting bilinear problem as a bipartite bilinear program (BBP). In other words, BBP is an optimization problem of the following form:

$$\begin{array}{ll} \min & x^\top Q_0 y + d_1^\top x + d_2^\top y \\ \text{s.t.} & x^\top Q_k y + a_k^\top x + b_k^\top y + c_k = 0,\ k \in \{1,\dots,m\} \\ & l \le (x,y) \le u \\ & (x,y) \in \mathbb{R}^{n_1+n_2}, \end{array} \tag{1}$$

where $n_1, n_2 \in \mathbb{Z}_+,\ Q_0, Q_k \in \mathbb{R}^{n_1 \times n_2},\ d_1, a_k \in \mathbb{R}^{n_1},\ d_2, b_k \in \mathbb{R}^{n_2},\ c_k \in \mathbb{R},\ \forall k \in \{1,\dots,m\}$. The vectors $l, u \in \mathbb{R}^{n_1+n_2}$ define the box constraints on the decision variables and, without loss of generality, we assume that $l_i = 0,\ u_i = 1,\ \forall i \in \{1,\dots,n_1+n_2\}$. BBP (1) may include bipartite bilinear inequality constraints, which can be converted into equality constraints by adding slack variables, and these slack variables will also be bounded since the original variables are bounded.

We note that BBP is a special case of the more general biconvex optimization problem [15]. BBP has many applications such as waste water management [13, 8, 14], pooling problem [17, 19], and supply chain [29].

2 Our results

2.1 Second order cone representable relaxation of BBP

A common and successful approach in integer linear programing is to generate cutting-planes implied by single constraint relaxation, see for example [9, 25, 12, 4]. We take a similar approach here. We begin by examining one row relaxation of BBP, that is, we study the convex hull of the set defined by a single constraint defining the feasible region of (1). Our first result is to show that the convex hull of this set is second order cone (SOCP) representable in the extended space, where we have introduced new variables w_{ij} for $x_i y_j$. We formally present this result next.

Theorem 1. *Let $n_1, n_2 \in \mathbb{Z}_+$, $V_1 = \{1,\dots,n_1\}$, $V_2 = \{1,\dots,n_2\}$, and $E \subseteq V_1 \times V_2$. Consider the one-constraint BBP set*

$$S := \left\{ (x,y,w) \in [0,1]^{n_1+n_2+|E|} \;\middle|\; \begin{array}{l} \sum_{(i,j)\in E} q_{ij} w_{ij} + \sum_{i\in V_1} a_i x_i + \sum_{j \in V_2} b_j y_j + c = 0, \\ w_{ij} = x_i y_j,\ \forall (i,j) \in E \end{array} \right\}.$$

Then:

(i) Let $(\bar{x}, \bar{y}, \bar{w})$ be an extreme point of S. Then, there exists $U \subseteq V_1 \cup V_2$, of the form

- *(a) $U = \{i_0, j_0\}$ where $(i_0, j_0) \in E$, or*
- *(b) $U = \{i_0\}$ where $i_0 \in V_1$ is an isolated node, or*
- *(c) $U = \{j_0\}$ where $j_0 \in V_2$ is an isolated node,*

such that $\bar{x}_i \in \{0,1\},\ \forall i \in V_1 \setminus U$, and $\bar{y}_j \in \{0,1\},\ \forall j \in V_2 \setminus U$.

(ii) $\mathrm{conv}(S)$ *is SOCP-representable.*

A proof of Theorem 1 is presented in Section 3.1.

Remark 1 (Exponential size SOCP formulation)**.** *In Theorem 1, part (ii) follows from part (i). For any given choice of* U*, we first fix all the variables to* 0 *or* 1 *except for those in* U*. It is then shown that the convex hull of the resulting set is SOCP-representable and we obtain (ii) by convexifying the union of a finite set of SOCP representable sets.*

It is easy to see that the number of distinct U *sets is* $\mathcal{O}(n_1 n_2)$*, and the number of possible fixings is* $\mathcal{O}(2^{n_1+n_2})$*. Thus, the number of resulting SOCP representable objects is* $\mathcal{O}(n_1 n_2 2^{n_1+n_2})$*.*

We note that the literature in global optimization theory has many results on convexifying functions, see for example [1, 35, 26, 42, 43]. However, as is well-known, replacing a constraint $f(x) = b$ by $\{x \,|\, \hat{f}(x) \geq b,\ \breve{f}(x) \leq b\}$ where $\hat{f}$ and $\breve{f}$ are the concave and convex envelop of f, does not necessarily yield the convex hull of the set $\{x \,|\, f(x) = b\}$. There are relatively lesser number of results on convexification of sets [41, 30, 31, 40]. Theorem 1 generalizes results presented in [40, 16, 22] and is related to results presented in [34, 10].

The SOCP relaxation for the feasible region of the general BBP (1) that we propose, henceforth referred as S^{SOCP}, is the intersection of the convex hull of each of the constraints of (1). Formally:

$$S^{SOCP} = \bigcap_{k=1}^{m} \mathrm{conv}(S_k),$$

where $S_k = \{(x, y, w) \in [0,1]^{n_1 \times n_2 \times |E|} \,|\, x^\top Q_k y + a_k^\top x + b_k^\top y + c_k = 0, w_{ij} = x_i y_j \ \forall (i,j) \in E\}$ and E is the edge set of the graph corresponding to the BBP instance (and not just of one row). As an aside, note that S^{SOCP} can be further strengthened by adding the convex hull of single row BBP sets arrived by taking linear combinations of rows.

Next we discuss the strength of S^{SOCP} vis-á-vis the strength of other standard relaxations. Consider the following two standard relaxations of the feasible region of BBP (1):

- Let S^{SDP} be the standard semi-definite programming (SDP) relaxation: Let H be the matrix variable representing $\begin{bmatrix} x \\ y \end{bmatrix} [x^\top y^\top]$. We write $w = \mathrm{proj}_E(H)$, to imply that if $(i,j) \in E$, then $w_{ij} = \frac{1}{2}\left(H_{i(j+n_1)} + H_{(j+n_1)i}\right)$. Then the feasible region of the standard SDP relaxation, S^{SDP}, may be written as:

$$\sum_{ij \in E} (Q_k)_{ij} w_{ij} + a_k^\top x + b_k^\top y + c_k = 0,\ k \in \{1, \ldots, m\} \tag{2}$$

$$\mathrm{proj}_E(H) = w \tag{3}$$

$$\begin{bmatrix} H & [x^\top y^\top] \\ \begin{bmatrix} x \\ y \end{bmatrix} & 1 \end{bmatrix} \succeq 0. \tag{4}$$

- Let

$$S^{QBP} := \{(x,y,w) \in [0,1]^{n_1+n_2+|E|} \mid (2)\} \bigcap \text{conv}\left(\{(x,y,w) \in [0,1]^{n_1+n_2+|E|} \mid w_{ij} = x_i y_j \ \forall (i,j) \in E\}\right). \quad (5)$$

  Note that S^{QBP} is a polyhedral set, since the second set in the right-hand-side of (5) is equal to the Boolean Quadratic Polytope [7]. Two well-known classes of valid inequalities for this set are the McCormick's inequalities [1] and the triangle inequalities [33].

Theorem 2. *For any BBP, we have that*

$$\text{proj}_{x,y,w}\left(S^{SDP}\right) \bigcap S^{QBP} \supseteq S^{SOCP}.$$

A proof of Theorem 2 is presented in Section 3.2.

Remark 2 (Importance of the extended variables w)**.** *It is possible to show that the convex hull of one row BBP is SOCP representable, even without introducing the w variables. Thus, it is possible to construct, similar to S^{SOCP}, a SOCP-representable relaxation of BBP, without introducing w variables. However, this SOCP relaxation would be weaker. In particular, we are unable to prove the corresponding version of Theorem 2 for this SOCP relaxation. The strength of S^{SOCP} relaxation is due to the fact that the extended space w variables 'interact' from different constraints.*

We note that other SOCP relaxations for QCQPs have been proposed [21, 6]. However, these are all weaker than the standard SDP relaxation.

We also note that it is polynomial time to optimize on S^{SDP}, however, using current software, SOCPs can be solved much faster than SDPs. It is NP-hard to optimize on S^{QBP}, although, as discussed in Remark 1, the size of the extended formulation to obtain S^{SOCP} is exponential in size.

Finally, we note that any QCQP–even if it has other type of constraints–could have its standard relaxation (for example McCormick or SDP) strengthened by using the convexification technique presented here. In particular, any quadratic constraint can be converted into a bipartite bilinear constraint by introducing auxiliary variables. In general, given any quadratic constraint in n variables, it is always possible to introduce at most n auxillary variables to obtain a BBP relaxation. As an example:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_i x_j + a^\top x + c = 0 \quad (6)$$

may be equivalently written as:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_i y_j + a^\top x + c = 0 \quad (7)$$
$$x_j = y_j \forall j \in [n].$$

Now one can apply the convexification to the set described by equation (7) together with bound constraints.

2.2 A new branching rule

For details about general branch-and-bound scheme for global optimization see, for example, [36]. Inspired by the convex relaxation described in Section 2.1, we propose a new rule for partitioning the domain of a given variable in order to produce two branches. Details of this new proposed branching rule together with node selection and variable selection rules that we used in our computational experiments are presented in Section 4.

Here, we sketch the main ideas behind our new proposed branching rule. Suppose we have decided to branch on the variable x_1. As explained in Remark 1, the convex hull of the one constraint set is obtained by taking the convex hull of union of sets obtained by fixing all but two (or one) variables. If we are branching on x_1, we examine all such two-variable sets involving x_1 obtained from each of the constraints. For each of these sets, there is an ideal point to divide the range of x_1 so that the sum of the volume of the two convex hulls of the two-dimensional sets corresponding to the two resulting branches is minimized. (See recent papers on importance of volume minimization in branch-and-bound algorithm [38]). We present a heuristic to find an "ideal range". We collect all such ideal ranges corresponding to all the two-dimensional sets involving x_1. Then we present a heuristic to select one points (based on corresponding volume reduction) to finally partition the domain of x_1. We also use similar arguments to propose a new variable selection rule.

2.3 A new application of BBP

A new application of BBP, which motivated our work presented here, is called as the *finite element model updating problem*, which is a fundamental methodological problem in structural engineering. See Section 5.1 for a description of the problem. All the new methods we develop here are tested on instances of this problem.

2.4 Techniques to make the SOCP relaxation tractable

Due to the large size of S^{SOCP}, in practice, we consider a lighter version of this relaxation. In particular, we write the extended formulation of each row of BBP corresponding only to the variables in that row (see details in Section 5.2.1). As our instances are row sparse, the resulting SOCP relaxation can be solved in reasonable time. Unfortunately, there are no theoretical guarantees for the bounds of this light version of the relaxation.

After some preliminary experimentation, we observed that a polyhedral outer approximation of the SOCP relaxation produces similar bounds but solves much faster. Therefore, we used this linear programming (LP) relaxation in our experiments. Details of this outer approximation are presented in Section 5.2.2. We emphasise here that, this outer approximation is crucial for computational success.

2.5 Computational Experiments

Our computational experiments are aimed at making three comparisons. First, we examined the quality of the dual bound produced at root node via our new method (polyhedral outer

approximation of SOCP relaxation) against SDP, McCormick, and SDP together with McCormick inequalities. The bounds produced are better for the new method. Second, we test the performance of the new branching rule against traditional branching rules. Our experiments show that the new branching rule significantly outperforms the other branching rules. Finally, we compare the performance of our naive branch-and-bound implementation against BARON. In all instances, we close significantly more gap in equal amount of time. All these results are discussed in detail in Section 5.3.

3 Second order cone representable relaxation and its strength

3.1 Proof of Theorem 1

Consider the bipartite graph $G = (V_1, V_2, E)$ defined by the set of vertices $V_1 = \{1, \dots, n_1\}$ and $V_2 = \{1, \dots, n_2\}$ which is associated to the equation

$$\sum_{(i,j)\in E} q_{ij} x_i y_j + \sum_{i\in V_1} a_i x_i + \sum_{j\in V_2} b_j y_j + c = 0. \tag{EQ}$$

In this section, we prove that the convex hull of the set

$$S = \{(x, y, w) \in [0,1]^{n_1+n_2+|E|} \,|\, (EQ),\ w_{ij} = x_i y_j\ \forall (i,j) \in E\}. \tag{8}$$

is SOCP representable. In addition, the proof provides an implementable procedure to obtain $\text{conv}(S)$. The key idea underlying this result is the fact that, at each extreme point of S, at most two variables are not fixed to 0 or 1 and, once all variables but two (or one) are fixed, the convex hull of the resulting object is SOCP representable in $\mathbb{R}^2$ (or $\mathbb{R}$). Hence, $\text{conv}(S)$ can be written as the convex hull of a union of SOCP representable sets.

3.1.1 Preliminary results

First we present a few preliminary results that will be used to prove that $\text{conv}(S)$ is SOCP representable.

Lemma 1. *[39] Let $f : [0,1]^n \to \mathbb{R}$ be a continuous function and $B \subseteq [0,1]^n$ be a convex set. Then*

$$\text{conv}(\{x \in B \,|\, f(x) = 0\}) = \text{conv}\left(\{x \in B \,|\, f(x) \leq 0\}\right) \bigcap \text{conv}(\{x \in B \,|\, f(x) \geq 0\}).$$

Lemma 2. *[20] Let $f : [0,1]^n \to \mathbb{R}$ be a convex function. Then*

$$G := \text{conv}(\{x \in [0,1]^n \,|\, f(x) \geq 0\}),$$

is a polytope. Indeed, G can be obtained as the convex hull of finite number of points obtained as follows: fix all but one variable to 0 or 1 and solve for $f(x) = 0$.

Lemma 3. *[3] Let $T \subset \mathbb{R}^n$ be a compact set and $\{T_k\}_{k\in K}$ be a partition of the set of all extreme points of T. Then,*

$$\operatorname{conv}(T) = \operatorname{conv}\left(\bigcup_{k\in K} T_k\right) = \operatorname{conv}\left(\bigcup_{k\in K} \operatorname{conv}(T_k)\right). \tag{9}$$

In addition, if $\operatorname{conv}(T_k)$ is a SOCP representable set for every $k \in K$, then $\operatorname{conv}(T)$ is also a SOCP representable set.

Lemma 4. *Let $B = \{(x,w) \in [0,1]^n \times \mathbb{R} \,|\, x \in B_0,\ w = l^\top x + l_0\}$, where $B_0 \subseteq \mathbb{R}^n$, and $l^\top x + l_0$ is an affine function of x. Then,*

$$\operatorname{conv}(B) = \{(x,w) \in [0,1]^n \times \mathbb{R} \,|\, x \in \operatorname{conv}(B_0),\ w = l^\top x + l_0\}.$$

Proof. We assume B_0 is non-empty, otherwise, there is nothing to prove. Let $(x,w) \in \operatorname{conv}(B)$. Then there exist $(x^i, w^i) \in B$ and $\lambda_i \geq 0,\ \forall i \in \{1,\ldots,n+2\}$, such that $\sum_{i=1}^{n+2} \lambda_i = 1$, $x = \sum_{i=1}^{n+2} \lambda_i x^i$ and $w = \sum_{i=1}^{n+2} \lambda_i w^i$. It follows by the definition of B that $x^i \in B_0,\ \forall i \in \{1,\ldots,n+2\}$, and hence $x \in \operatorname{conv}(B_0)$. It also follows from the definition of B that $w^i = l^\top x^i + l_0,\ \forall i \in \{1,\ldots,n+2\}$, and hence

$$w = \sum_{i=1}^{n+2} \lambda_i w^i = \sum_{i=1}^{n+2} \lambda_i (l^\top x^i + l_0) = l^\top \left(\sum_{i=1}^{n+2} \lambda_i x^i\right) + l_0 = l^\top x + l_0.$$

Conversely, let (x,w) be such that $x \in \operatorname{conv}(B_0)$ and $w = l^\top x + l_0$. Then, there exist $x^i \in B_0$ and $\lambda_i \geq 0,\ \forall i \in \{1,\ldots,n+1\}$, such that $\sum_{i=1}^{n+1} \lambda_i = 1$, $x = \sum_{i=1}^{n+1} \lambda_i x^i$. Define $w^i = l^\top x^i + l_0,\ \forall i \in \{1,\ldots,n+1\}$. Then $(x^i, w^i) \in B,\ \forall i \in \{1,\ldots,n+1\}$. In addition,

$$w = l^\top x + l_0 = l^\top \left(\sum_{i=1}^{n+1} \lambda_i x^i\right) + l_0 = \sum_{i=1}^{n+1} \lambda_i (l^\top x^i + l_0) = \sum_{i=1}^{n+1} \lambda_i w^i,$$

which completes the proof. □

3.1.2 Proof of part (i) of Theorem 1

We restate part (i) of Theorem 1 next for easy reference:

Proposition 1. *Let $(\bar{x}, \bar{y}, \bar{w})$ be an extreme point of the set S defined in (8). Then, there exists $U \subseteq V_1 \cup V_2$, of the form*

1. *$U = \{i_0, j_0\}$ where $(i_0, j_0) \in E$, or,*
2. *$U = \{i_0\}$ where $i_0 \in V_1$ is an isolated node, or,*
3. *$U = \{j_0\}$ where $j_0 \in V_2$ is an isolated node,*

such that $\bar{x}_i \in \{0,1\},\ \forall i \in V_1 \setminus U$, and $\bar{y}_j,\ \forall j \in V_2 \setminus U$.

Proof. To prove by contradiction, suppose without loss of generality that $0 < \bar{x}_1, \bar{x}_2 < 1$. Consider the system of equations

$$\begin{aligned} \bar{a}_1 x_1 + \bar{a}_2 x_2 + \bar{c} &= 0, \\ w_{1j} - x_1 \bar{y}_j &= 0 \ \forall j : (1,j) \in E \\ w_{2j} - x_2 \bar{y}_j &= 0 \ \forall j : (2,j) \in E, \end{aligned}$$

obtained by fixing $x_i = \bar{x}_i,\ y_j = \bar{y}_j$ in (8), $w_{ij} = \bar{x}_i \bar{y}_j\ \forall i \in V_1 \setminus \{1,2\},\ \forall j \in V_2$. Since $(\bar{x}_1, \bar{x}_2)$ is in the relative interior of $\{(x_1, x_2) \in [0,1]^2 \,|\, \bar{a}_1 x_1 + \bar{a}_2 x_2 + \bar{c} = 0\}$, $(\bar{x}, \bar{y}, \bar{w})$ cannot be an extreme point of S. □

3.1.3 Proof of part (ii) of Theorem 1

First, we prove that the two-variable sets we encounter after fixing variables are SOCP representable.

Proposition 2. *Let* $S_0 = \{(x,y) \in [0,1]^2 \,|\ ax + by + qxy + c = 0\}$. *Then,* $\mathrm{conv}(S_0)$ *is SOCP representable.*

Proof. We may assume $S_0 \neq \emptyset$ and $q \neq 0$, otherwise the result follows trivially. Define $r = -b/q,\ s = -a/q$ and $\tau = (ab - cq)/q^2$ to write $ax + by + qxy + c = 0$ equivalently as

$$(x - r)(y - s) = \tau. \tag{10}$$

If $\tau = 0$, then (10) is equivalent to $x = r$ or $y = s$. In this case, $S_0 = \{(x,y) \in [0,1]^2 \,|\, x = r\} \cup \{(x,y) \in [0,1]^2 \,|\, y = s\}$ and hence $\mathrm{conv}(S_0)$ is a polytope. Suppose $\tau > 0$ (if $\tau < 0$, we multiply (10) by -1 and repeat the same proof with $x - r$ and τ replaced with $-(x - r)$ and $-\tau$). Either $x - r, y - s \geq 0$ or $x - r, y - s \leq 0$. Thus, $S_0 = S_0^{>} \cup S_0^{<}$, where $S_0^{>} = \{(x,y) \in [0,1]^2 \,|\, x - r, y - s \geq 0,\ (10)\}$ and $S_0^{<} = \{(x,y) \in [0,1]^2 \,|\, x - r, y - s \leq 0,\ (10)\}$. Next, we show that if $S_0^{>} \neq \emptyset$, then $\mathrm{conv}(S_0^{>})$ is SOCP representable. Using that $4uv = (u+v)^2 - (u-v)^2$, we can rewrite (10) as

$$\sqrt{[(x - r) - (y - s)]^2 + (2\sqrt{\tau})^2} = (x - r) + (y - s).$$

It now follows from Lemma 1 that $\mathrm{conv}(S_0^{>}) = \mathrm{conv}(S_1^{>}) \cap \mathrm{conv}(S_2^{>})$, where

$$S_1^{>} = \{(x,y) \in [0,1]^2 \,|\, x - r, y - s \geq 0,\ \sqrt{[(x - r) - (y - s)]^2 + (2\sqrt{\tau})^2} \leq (x - r) + (y - s)\}$$
$$S_2^{>} = \{(x,y) \in [0,1]^2 \,|\, x - r, y - s \geq 0,\ \sqrt{[(x - r) - (y - s)]^2 + (2\sqrt{\tau})^2} \geq (x - r) + (y - s)\}.$$

Notice that $S_1^{>}$ is SOCP representable. Also, as the square root term in the definition of $S_2^{>}$ is a convex function in x and y, it follows from Lemma 2 that $S_2^{>}$ is a polytope. Thus, $\mathrm{conv}(S_0^{>})$ is SOCP representable. Similarly, we can prove that $\mathrm{conv}(S_0^{<})$ is SOCP representable by repeating the arguments above after replacing $x - r, y - s$ with $-(x - r), -(y - s)$. Therefore, $\mathrm{conv}(S_0) = \mathrm{conv}(S_0^{>} \cup S_0^{<}) = \mathrm{conv}(\mathrm{conv}(S_0^{>}) \cup \mathrm{conv}(S_0^{<}))$ is SOCP representable by Lemma 3. □

Proposition 3. *Let* $S_0 = \{(x,y) \in [0,1]^2 \mid \ y = a_0 + a_1 x + a_2 x^2\}$. *Then* $\mathrm{conv}(S_0)$ *is SOCP representable.*

Proof. We may assume $S_0 \neq \emptyset$ and $a_2 \neq 0$, otherwise the result follows trivially. By completing squares, we can write $y = a_0 + a_1 x + a_2 x^2$ equivalently as $(x + 0.5a_1/a_2)^2 - (a_1/2a_2)^2 + a_0/a_2 = y/a_2$, and then as

$$(x+\bar{a})^2 = t \ \Leftrightarrow \ \sqrt{(x+\bar{a})^2 + \left(\frac{t-1}{2}\right)^2} = \frac{t+1}{2}, \tag{11}$$

where $\bar{a} = 0.5a_1/a_2$, $t = y/a_2 + (a_1/2a_2)^2 - a_0/a_2$, using that $4t = (t+1)^2 - (t-1)^2$. It now follows from Lemma 1 that $\mathrm{conv}(S_0) = \mathrm{conv}(S_1) \cap \mathrm{conv}(S_2)$, where

$$S_1 = \{(x,y) \in [0,1]^2 \mid \sqrt{(x+\bar{a})^2 + \left(\frac{t-1}{2}\right)^2} \leq \frac{t+1}{2}\}$$

$$S_2 = \{(x,y) \in [0,1]^2 \mid \sqrt{(x+\bar{a})^2 + \left(\frac{t-1}{2}\right)^2} \geq \frac{t+1}{2}\}.$$

Notice that S_1 is SOCP representable. Also, as the square root term in the definition of S_2 is a convex function in x and y (because t is an affine function of y), it follows from Lemma 2 that S_2 is a polytope. Thus, $\mathrm{conv}(S_0)$ is SOCP representable. □

Proposition 4. *Let* $S_0 = \{(x,y,w) \in [0,1]^3 \mid ax + by + qw + c = 0, \ w = xy\}$. *Then,* $\mathrm{conv}(S_0)$ *is SOCP representable.*

Proof. If $q \neq 0$, then we can write

$$S_0 = \{(x,y,w) \in [0,1]^2 \times \mathbb{R} \mid (x,y) \in B_0, \ w = (-c - ax - by)/q\}, \tag{12}$$

where $B_0 = \{(x,y) \in [0,1]^2 \mid ax + by + qxy + c = 0\}$. (Note that the bounds on w are automatically enforced in (12) and it is sufficient to say $w \in \mathbb{R}$). Hence, by Proposition 2 and Lemma 4, $\mathrm{conv}(S_0)$ is SOCP representable.

Now, suppose $q = 0$. Four cases: (i) $a, b = 0$. In this case, we may assume $c = 0$, otherwise $S_0 = \emptyset$. Then, $S_0 = \{(x,y,w) \in [0,1]^3 \mid w = xy\}$, in which case $\mathrm{conv}(S_0)$ is a well known polytope given by the McCormick envelope. (ii) $a = 0$, $b \neq 0$. In this case, if $-c/b \notin [0,1]$, then S_0 is infeasible. Otherwise, this case is trivial. (iii) $a \neq 0$, $b = 0$. Similar to previous case. (iv) $a \neq 0$ and $b \neq 0$. In this case, we can solve $ax + by + c = 0$ for x, i.e. $x = (-c - by)/a$. Let $[\alpha, \beta]$ be the bounds on y such that the line $ax + by + c = 0$ intersects the $[0,1]^2$ box. If $\alpha = \beta$, then we can set $y = \alpha$ and the result follows trivially. Otherwise, substitute in $w = xy$ to rewrite S_0 as following

$$S_0 = \{(x,y,w) \in \mathbb{R} \times [\alpha,\beta] \times [0,1] \mid (y,w) \in B_0, \ x = (-by - c)/a\},$$

where $B_0 = \{(y,w) \in [\alpha,\beta] \times [0,1] \mid w = (-c/a)y - (b/a)y^2\}$. Now, it is straightforward via Proposition 3 (affinely scale y to have bound of $[0,1]$) and Lemma 4 that $\mathrm{conv}(S_0)$ is a SOCP representable set. □

Now we are ready to prove part (ii) of Theorem 1.

Proposition 5. *Let S be the set defined in (8). Then* $\text{conv}(S)$ *is SOCP representable.*

Proof. By Proposition 1, we can fix various sets of x and y variables that corresponds to the U sets and prove that the convex hull of each of these sets is SOCP representable. Case (i): $|U| = 1$. In this case, the set of unfixed variables satisfy a set of linear equations. Thus this set is clearly SOCP representable. Case (ii): $U = \{(i_0, j_0)\}$, where $(i_0, j_0) \in E$. In this case, the set of unfixed variables satisfy the following constraints:

$$\begin{aligned} ax_{i_0} + by_{j_0} + qw_{i_0j_0} + c &= 0, && (13)\\ w_{i_0j_0} &= x_{i_0}y_{j_0} && (14)\\ w_{ij_0} &= \bar{x}_i y_{j_0}\ \forall (i, j_0) \in E, i \neq i_0 && (15)\\ w_{i_0j} &= \bar{y}_j x_{i_0}\ \forall (i_0, j) \in E, j \neq j_0, && (16) \end{aligned}$$

where the bound constraints on w_{ij_0} and w_{i_0j} variables are not needed explictly. Thus, by Proposition 4 and Lemma 4, the above set is SOCP representable. Thus, by Lemma 3, we obtain that $\text{conv}(S)$ is SOCP representable. □

3.2 Proof of Theorem 2

In order to prove Theorem 2 it is sufficient to prove that:

$$\text{proj}_{x,y,w}\left(S^{SDP}\right) \supseteq S^{SOCP} \tag{17}$$

and

$$S^{QBP} \supseteq S^{SOCP}. \tag{18}$$

We prove these two containments next.

Proposition 6. *For any BBP, (17) holds.*

Proof. Let

$$T^k := \{(x, y, H, w) \,|\, (2) \text{ corresponding to } k, (3), \text{ and } (4)\}$$

and as before let

$$S^k := \{(x, y, w) \,|\, (2) \text{ corresponding to } k, w_{ij} = x_i y_j\ \forall (ij) \in E\}.$$

Then by construction

$$\text{proj}_{x,y,w}\left(T^k\right) \supseteq \text{conv}(S^k). \tag{19}$$

Next we need the following:

Claim 1 $\bigcap_{k=1}^{m} \mathrm{proj}_{x,y,w}\left(T^k\right) = \mathrm{proj}_{x,y,w}\left(\bigcap_{k=1}^{m}\left(T^k\right)\right)$: Trivially we have that,

$$\bigcap_{k=1}^{m} \mathrm{proj}_{x,y,w}\left(T^k\right) \supseteq \mathrm{proj}_{x,y,w}\left(\bigcap_{k=1}^{m}\left(T^k\right)\right),$$

holds.

We now verify the converse. For some $(\bar{x},\bar{y},\bar{w}) \in \mathrm{proj}_{x,y,w}\left(T^k\right)$, let

$$\mathcal{H}^k(\bar{x},\bar{y},\bar{w}) := \left\{H \,|\, (\bar{x},\bar{y},\bar{w},H) \in T^k\right\}.$$

Then observe that $\mathcal{H}^k(\bar{x},\bar{y},\bar{w})$ is the set of matrices H satisfying

$$\mathrm{proj}_E(H) = \bar{w} \tag{20}$$

$$\begin{bmatrix} H & \begin{bmatrix}\bar{x}^\top \bar{y}^\top\end{bmatrix} \\ \begin{bmatrix}\bar{x}\\ \bar{y}\end{bmatrix} & 1 \end{bmatrix} \succeq 0. \tag{21}$$

Thus $\mathcal{H}^k(\bar{x},\bar{y},\bar{w})$ is independent of k, i.e. if $(\bar{x},\bar{y},\bar{w}) \in \bigcap_{k=1}^{m} \mathrm{proj}_{x,y,w}\left(T^k\right)$ then $\mathcal{H}^{k_1}(\bar{x},\bar{y},\bar{w}) = \mathcal{H}^{k_2}(\bar{x},\bar{y},\bar{w})$ for all $k_1 \neq k_2$. Therefore in particular, if $(\bar{x},\bar{y},\bar{w}) \in \bigcap_{k=1}^{m} \mathrm{proj}_{x,y,w}\left(T^k\right)$, then there exists $\bar{H}$ such that $(\bar{x},\bar{y},\bar{w},\bar{H}) \in \bigcap_{k=1}^{m} T^k$. Thus, $(\bar{x},\bar{y},\bar{w}) \in \mathrm{proj}_{x,y,w}\left(\bigcap_{k=1}^{m} T^k\right)$. $\diamond$

Now, we return to the proof of the original statement. Intersecting (19) for all $k \in \{1,\ldots,m\}$ we obtain,

$$\mathrm{proj}_{x,y,w}\left(S^{SDP}\right) = \mathrm{proj}_{x,y,w}\left(\bigcap_{k=1}^{m}\left(T^k\right)\right) = \bigcap_{k=1}^{m} \mathrm{proj}_{x,y,w}\left(T^k\right) \supseteq \bigcap_{k=1}^{m} \mathrm{conv}(S^k) = S^{SOCP},$$

where the first equality is by definition of S^{SDP}, the second equality via Claim 1, the inequality is due to (19) and the last equality is by definition of S^{SOCP}. $\square$

Proposition 7. *For any BBP, (18) holds.*

Proof. Recall that S^{QBP} is the set

$$\left\{(x,y,w) \in [0,1]^{n_1+n_2+|E|} \,|\, \sum_{(ij)\in E} (Q_k)_{ij} w_{ij} + a_k^\top x + b_k^\top y + c_k = 0 \;\; k \in \{1,\ldots,m\}\right\} \tag{22}$$

$$\bigcap \mathrm{conv}\left(\{(x,y,w) \in [0,1]^{n_1+n_2+|E|} \,|\, w_{ij} = x_i y_j \;\forall (i,j) \in E\}\right). \tag{23}$$

Let

$$T^k := \{(x,y,w) \in [0,1]^{n_1+n_2+|E|} \,|\, (22) \text{ corresponding to } k, (23)\}$$

and let

$$S^k := \{(x,y,w) \,|\, (2) \text{ corresponding to } k, w_{ij} = x_i y_j \;\forall (ij) \in E\}.$$

Then by construction

$$T^k \supseteq \text{conv}(S^k). \tag{24}$$

Intersecting (24) for all $k \in \{1, \ldots, m\}$ we obtain,

$$S^{QBP} = \bigcap_{k=1}^{m} T^k \supseteq \bigcap_{k=1}^{m} \text{conv}(S^k) = S^{SOCP}.$$

□

4 Proposed branch-and-bound algorithm

In this section, we discuss some details of our proposed branch-and-bound algorithm to solve BBP (1).

4.1 Node selection and partitioning strategies

A common node selection rule used in the literature is the so-called *best-bound-first*, in which a node with the least lower bound (assuming minimization) is chosen for branching. Other rules may include selection of nodes that have the potential of identifying good feasible solutions earlier. In our computational experiments, we only use best-bound-first rule. Also, we use the most simple partitioning operation: rectangular. Examples of other operation adopted in the literature are conical and simplicial [23].

4.2 Variable selection and point of partitioning

A simple rule for variable selection is to choose a variable with largest range. Another common rule is to prioritize the variable that is most responsive for the approximation error of nonlinear terms. For example, suppose we are optimizing in the extended space of (x, y, w), then we could chose x_i (or y_j) for which the absolute error $|\bar{w}_{ij} - \bar{x}_i \bar{y}_j|$ is maximized over the set of all possible pairs (i, j), where $(\bar{x}, \bar{y}, \bar{w})$ is the relaxation solution for the current node. We refer to this rule as the *gap-error-rule*.

Once the variable is selected, say x_1 (without loss of generality), we can list three standard rules for choosing the partitioning point:
Bisection: partition at the mid point of the domain of x_1 in the current node.
Maximum-deviation: partition at $\bar{x}_1$, where $(\bar{x}, \bar{y}, \bar{w})$ is the relaxation solution for the current node.
Incumbent: partition at x_1^*, where (x^*, y^*, w^*) is the current best feasible solution, if x_1^* is in the range of x_1 in the current node.

Combinations of the above rules have also been proposed. For example, Tawarmalani et al. [37] propose a rule that is a convex combination of bisection and maximum-deviation branching

rules (biased towards the maximum-deviation), and uses incumbent branching whenever possible. For more on branching rules, including strategies adopted by some of the state-of-the-art solvers for MINLP, see [28, 2, 44].

In our proposed algorithm, we use specialized variable and branching point selection rules, which use information collected from multiple disjunctions and, therefore, take into account the coefficients of the constraints in the model in addition to the variable ranges at the current node.

New proposed rule Note that we always branch on only one set of variables, either x or y. We describe our rule assuming we are branching on the x variables. To further ease exposition, we explain our proposed branching rules for the root node, i.e., we assume that all variables range from 0 to 1. Consider the three-variable set:

$$S_0 = \{(x_1, y_1, w_{11}) \in \mathbb{R}^3 \,|\, qw_{11} + ax_1 + by_1 + c = 0,\ w_{11} = x_1 y_1\},$$

which is obtained by fixing x_i, y_j to either 0 or 1 in (EQ), $\forall i \in V_1 \setminus \{1\},\ \forall j \in V_2 \setminus \{1\}$. Like the proof of Proposition 4, there are two cases of interest.

- $q \neq 0$. In this case, w_{11} can be written as affine function of x_1 and y_1. We can then write the projection of S_0 in the space of (x_1, y_1) as (we drop the indices to simplify notation, we also drop the word 'Proj')

  $$S_0 = \{(x, y) \in [0,1]^2 \,|\ (x - r)(y - s) = \tau\},$$

  where r, s, τ are constants. The equation $(x - r)(y - s) = \tau$ represents a hyperbola with asymptotes $x = r$ and $y = s$. Two typical instances are plotted in Figure 1-2, where the continuous thick portion of the curves represents S_0 and the whole dotted areas represent $\text{conv}(S_0)$. Our goal is to branch at a point that maximizes the eliminated area upon branching.

  Case 1: Both branches of a hyperbola intersect with the $[0,1]^2$ box. Let x_l (resp. x_u) be the x-coordinate of the intersection point of the left (resp. right) branch with either of the lines $y = 0$ or $y = 1$. The plot on Figure 1 suggests that branching x at any point $x_0 \in [x_l, x_u]$ is a reasonable choice for the case where both branches of the hyperbola intersect the $[0,1]^2$ box. Indeed, such branching would eliminate the entire dotted area between the two branches of the curve.

  Case 2: One branch of hyperbola intersects with the $[0,1]^2$ box. For the case where only one branch intersects the $[0,1]^2$ box, as illustrated in Figure 2, we could in principle compute C that maximizes the area of the triangle $\triangle_{ABC}$. To simplify the rule and avoid excessive computations, we simply choose C to be the point at which the tangent line to the curve is parallel to the line AB. Moreover, for points in some interval $[x_l, x_u]$ containing x_c, the area of the triangle $\triangle_{ABC}$ does not change much, implying that every point in $[x_l, x_u]$ may be a good choice to branch at. In our computational experiments, we compute x_l and x_u such that $x_c - x_l = \gamma(x_c - x_a)$ and $x_u - x_c = \gamma(x_b - x_c)$ with $\gamma = 2/3$.

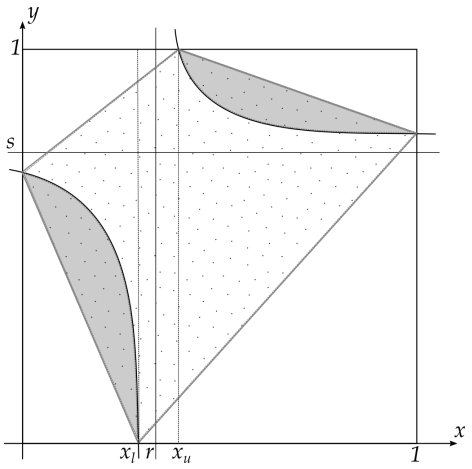

Figure 1: Convex hull of the set defined by the intersection of two branches of a hyperbola with the $[0,1]^2$ box. Here, x_l (resp. x_u) is the x-coordinate of the intersection point of the left (resp. right) branch with the line $y = 0$ (resp. $y = 1$).

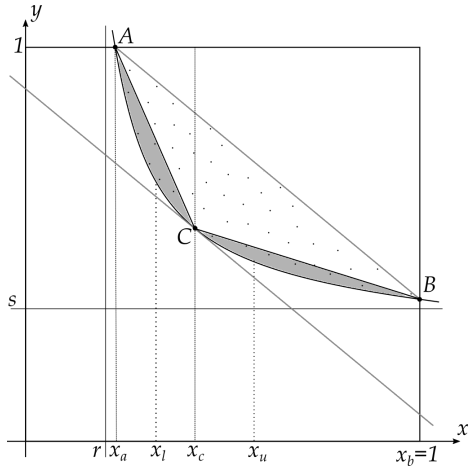

Figure 2: Convex hull of the set defined by the intersection of a single branch of a hyperbola with the $[0,1]^2$ box. Let A and B are the intersection points of the curve with the $[0,1]^2$ box and C is the point of the curve at which the tangent line is parallel to AB. Then, x_a, x_b and x_c are the projections of A, B and C onto the x axis.

- $q = 0$ and $a \neq 0$ or $b \neq 0$. Without loss of generality assume $b \neq 0$. In this case, y_1 is an affine function of x_1 as shown in proof of Proposition 4. Thus, we can study S_0 in the space of (x_1, w_{11}), where it is defined by a parabola and we adopt the same rule defined for the case of Figure 2, i.e. choose points x_l and x_u as a function of x_a and x_b. If the parabola intersects the $[0,1]^2$ box in more than two points, we define A and B to be the left and right most intersection points.

  Note that if $a \neq 0$, then x_1 is an affine function or y_1. We can identify appropriate points in the y_1 space as above and then translate them to the x space via the affine function.

Thus, corresponding to every three-variable set S_0, we associate (i) an x-variable x_i, (ii) an interval $[x_l, x_u]$ within the domain of x_i and (iii) we also approximately compute the area of $\text{conv}(S_0)$, (either in the space of (x_1, y_1), if $q \neq 0$, or in the space of (x_1, w_{11}), if $q = 0$), referred to as A_0. The actual area we use is that of the polyhedral outer approximation as will be discussed in Section 5.2.2.

Once the above data is collected for all disjunctions, we use Algorithm 1 to decide on the variable to branch on and the point of partitioning for this variable. In words, this algorithm integrates, in a discrete manner, over the approximation error coming from each disjunction. It starts by partitioning the domain of each x variable, and then estimates the error associated with each interval of that partition separately.

Finally, the algorithm selects the variable, and the corresponding interval from its partition, that minimizes the approximation error.

Algorithm 1 Branching rule

Input: $\delta = 1/K$, for some positive integer K. Let $\varepsilon_1, \varepsilon_2 > 0$.
Let $A_{ik} = 0,\ i \in \{1, \ldots, n_1\},\ k \in \{1, \ldots, K\}$. Let $p_i = 0,\ i \in \{1, \ldots, n_1\}$
Define $I_{ik} = [(k-1)\delta, k\delta]$, for $k \in \{1, \ldots, K\}$ (which defines a partition of the range of x_i).
for Each disjunctions S_0 **do**
    Compute (a) the index i of x-variable corresponding to S_0, (b) domain $[x_l, x_u]$ and (c) the area A_0.
    Set $p_i = p_i + 1$
    If $[x_l, x_u] \cap I_{ik} \neq \emptyset$ for some $k \in \{1, \ldots, K\}$, set $A_{ik} = A_{ik} + A_0$.
end for

for $i \in \{1, \ldots, n_1\}$ **do**
    if $\frac{p_i}{\sum_{l=1}^{n_1} p_l} < \varepsilon_1$ **then**
        variable i is declared irrelevant.
    end if
end for
Let $(i^*, k^*) \in \text{Argmax}\{A_{i,k} \,|\, i \in \{1, \ldots, n_1\}, i \text{ is not irrelevant}, k \in \{1, \ldots, K\}\}$
if $A_{i^*k^*} \geq \varepsilon_2$ **then**
    Branch on the variable x_{i^*} at the mid point of the interval $I_{i^*k^*}$.
else
    Use the bisection rule.
end if

In our computational experiments, whenever we use Algorithm 1, we set $\varepsilon_1 = 0.01,\ \varepsilon_2 = 1/16$ and $K = 8$. Our implementation is naive, and we have not tried to fine tune any of these parameters.

5 Computational experiments

5.1 Finite Element Updating Model

The instances of BBP that we use come from finite element (FE) model updating in structural engineering. The goal is to update the parameter values in an FE model, so that the model provides same resonance frequencies and mode shapes that are physically measured from vibration testing at the as-built structure. In this study we adopt the modal dynamic residual formulation, for which the details can be found in [45]. The formulation is briefly summarized as follows.

Consider the model updating of a structure with m number of degrees-of-freedom (DOFs). Corresponding to stiffness parameters that are being updated, the (scaled) updating variables are first denoted as $x \in [-1, 1]^{n_1}$. Since only some DOFs can be instrumented, we suppose n_2 of those are not instrumented, leaving $m - n_2$ of them as instrumented. In the meantime, it's assumed that n_3 number of vibration modes are measured/observed from the vibration testing

data. For each l-th measured mode, $\forall l \in \{1,\ldots,n_3\}$, the experimental results provide λ_l as the square of the (angular) resonance frequency, and $\bar{y}^l \in \mathbb{R}^{m-n_2}$ as the mode shape entries at the instrumented DOFs. In mathematical terms, the modal dynamic residual formulation can be stated as the problem of simultaneously solving the following set of equations on stiffness updating variables $x \in [-1,1]^{n_1}$ and (scaled) unmeasured mode shape entries $y^l \in [-2,2]^{n_2}$, $\forall l \in \{1,\ldots,n_3\}$:

$$[K_0 + \sum_{i=1}^{n_1} x_i K_i - \lambda_l M] \begin{bmatrix} \bar{y}^l \\ y^l \end{bmatrix} = 0,\ l \in \{1,\ldots,n_3\}, \tag{25}$$

where $M, K_0, K_i \in \mathbb{R}^{m\times m}$, $\forall i \in \{1,\ldots,n_1\}$, $\lambda_l \in \mathbb{R}_+$ and $\bar{y}^l \in \mathbb{R}^{m-n_2}$, $\forall l \in \{1,\ldots,n_3\}$, are problem data. In practice, (25) is unlikely to have a feasible solution set of x and y^l, $l \in \{1,\ldots,n_3\}$, because of modeling and measurement inaccuracies. Therefore, we convert the problem of solving (25) into an optimization problem that aims to minimize the sum of the residuals, i.e., the absolute difference between left and right-hand-side of each equation. After some affine transformations and simplifications, this optimization problem can be stated as following:

$$\begin{aligned} \min\ & \sum_{k=1}^{m} z_k \\ \text{s.t.}\ & |x^\top Q_k y + a_k^\top x + b_k^\top y + c_k| = z_k,\ k \in \{1,\ldots,m\} \\ & x \in [0,1]^{n_1}, y \in [0,1]^{n_2}, \end{aligned} \tag{26}$$

where n_2 and m correspond to $n_2 n_3$ and $m n_3$, respectively, in the notation of (25). Finally, (26) is equivalent to the following BBP.

$$\begin{aligned} \min\ & \sum_{k=1}^{m} z'_k + z''_k \\ \text{s.t.}\ & x^\top Q_k y + a_k^\top x + b_k^\top y + c_k = z'_k - z''_k,\ k \in \{1,\ldots,m\} \\ & x \in [0,1]^{n_1}, y \in [0,1]^{n_2}. \\ & 0 \le z'_k, z''_k \le u,\ k \in \{1,\ldots,m\}. \end{aligned} \tag{27}$$

Instances:
The simulated structural example is similar to the planar truss structure in [45]. In order to simulate measurement noise, we add a normal-distributed random variable to the parameters λ^l and $\bar{y}^l$, $\forall l \in \{1,\ldots,n_3\}$, with mean zero and variance equal 2% of its actual value. In our case there are six modes, i.e. $n_3 = 6$. By taking different values for n_2, we then generate ten instances whose number of variables and constraints are given in Table 1.

The instances are available online at `https://www2.isye.gatech.edu/~sdey30/`.

Table 1: Instances description

Inst	# of x-variables	# of y-variables	# of equations	# of bilinear terms
inst1	6	180	312	990
inst2	6	180	312	954
inst3	6	168	312	966
inst4	6	168	312	972
inst5	6	156	312	900
inst6	6	144	312	780
inst7	6	132	312	756
inst8	6	132	312	756
inst9	6	120	312	684
inst10	6	120	312	684

5.2 Simplifying S^{SOCP}

5.2.1 A lighter version of S^{SOCP}

According to Remark 1, the number of disjunction needed to model the convex hull of a single bilinear equation can be computationally prohibitive for many instances of interest. To overcome this issue, in our computational experiments, we write the convex hull of each row only in the space of the variable appearing in it. In particular, for constraint k we work with $G(V^k, E^k)$, where V^k is the set of variables appearing in constraint k and E^k represent the complete bipartite graph between the x and y variables appearing in V^k. This possibly weaker relaxation is much more computationally cheaper that S^{SOCP} for our instances due to the sparsity on the coefficients of each bilinear equation. We denote this relaxation as $\text{light} - S^{SOCP}$.

5.2.2 Polyhedral outer approximation

As shown in Proposition 2 and Proposition 3, all the sets obtained after fixings are SOCP representable. Some are polyhedral while many of the others are not. Since linear programming techniques are more efficient and robust, than the non-linear counterpart, we approximate the non-polyhedral sets by a polyhedral set.

As shown in proof of Proposition 5, all the non-linear sets that we need to convexify in order to obtain the convex hull of the set S defined in (EQ) are of the form

$$S_{i_0j_0} = \{(x, y, w) \in [0,1]^{n_1+n_2+n_1n_2} \,|\, x_i, y_j \in \{0,1\},\ \forall i \in V_1 \setminus \{i_0\},\ \forall j \in V_2 \setminus \{j_0\},\\ \bar{q}w_{i_0j_0} + \bar{a}x_{i_0} + \bar{b}y_{i_0} + \bar{c} = 0,\ w_{ij} = x_iy_j,\ i \in V_1,\ j \in V_2\},$$

for some $(i_0, j_0) \in E$. Without loss of generality, suppose $i_0 = 1$ and $j_0 = 1$, in which case we want to outer approximate the following set $S_0 = \{(x_1, y_1, w_{11}) \in \mathbb{R}^3 \,|\, qw_1 + ax_1 + by_1 + c = 0,\ w_{11} = x_1y_1\}$. There are two cases of interest. The first case occurs when $q \neq 0$. In this case, w_{11} is an affine functions of x_1 and y_1 as following: $w_{11} = (-c - ax_1 - by_1)/q$;

$w_{1j} = x_1 y_j,\ \forall j \in \{1,\dots,n_2\}$; $w_{i1} = x_i y_1,\ \forall i \in \{1,\dots,n_1\}$; and $w_{ij} = x_i y_j,\ \forall i \in \{1,\dots,n_1\} \setminus \{1\},\ \forall j \in \{1,\dots,n_2\} \setminus \{1\}$. Hence, we only need to approximate $\text{conv}(S_0)$ in the space of (x_1, y_1). If both branches of the hyperbola defined by $qx_1y_1 + ax_1 + by_1 + c = 0$ intersect the $[0,1]^2$ box, than $\text{conv}(S_0)$ is polyhedral. Suppose only one branch of the hyperbola intersects the box. Then, we outer approximate $\text{conv}(S_0)$ by using tangent lines to the curve. In our implementation, we only use the tangent lines at the intersection points of the curve with the box, see Figure 3. More tangent lines could be added to better approximate $\text{conv}(S_0)$, but based on our preliminary experience on our instances it does not make significant difference.

The second case of interest is $q = 0$ and $a \neq 0$ (or $b \neq 0$) for which we can rewrite S_0 as $S_0 = \{(x_1, y_1, w_{11}) \in [0,1]^3 \,|\, aw_{11} = -by_1^2 - cy_1,\ ax_1 = -by_1 - c\}$. In this case, x_1 is an affine function of y_1 and we only need to approximate $\text{conv}(S_0)$ in the space of (y_1, w_{11}), where $aw_{11} = -cy_1 - by_1^2$ defines a parabola as shown in Figure 4. As in the previous case, we outer approximate the curve by using tangent lines to the curve as illustrated in Figure 4.

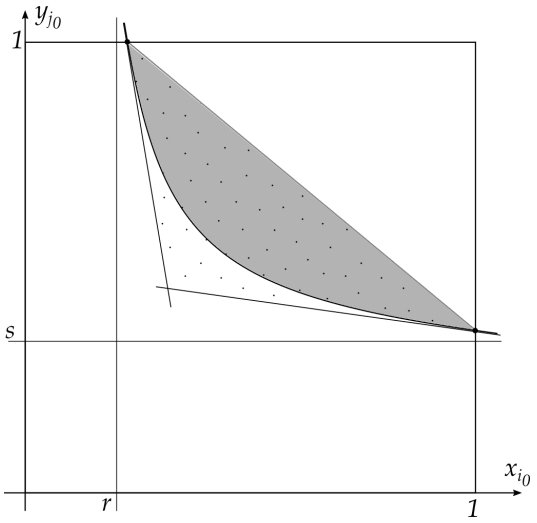

Figure 3: Convex hull of the set defined by the intersection of one branch of a hyperbola with the $[0,1]^2$ box, and its tangential linear outer approximation.

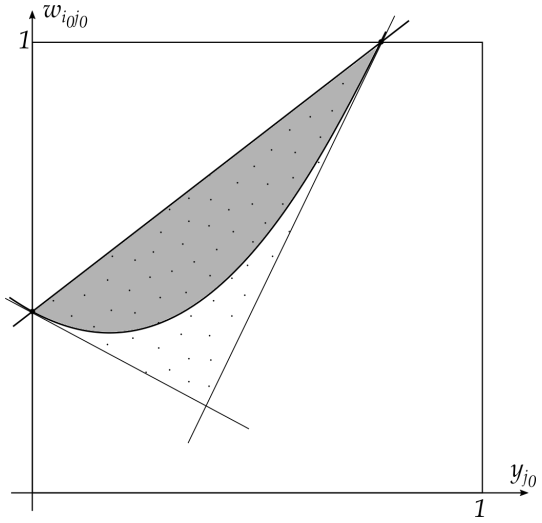

Figure 4: Convex hull of the set defined by the intersection of parabola with the $[0,1]^2$ box, and its tangential linear outer approximation.

We note that each polyhedral set obtained this way, is defined by a few extreme points. Therefore, when writing the disjunctive formulation, we use these extreme points rather then the linear decription of each polyhedral set.

5.3 Computation results

5.3.1 Software and Hardware

All of our experiments were run on a Windows 10 machine with 64-bit operating system, x64 based processor with 2.19GHz, and 32GB RAM. We call MOSEK via CVX from MATLAB

R2015b to solve SDPs. We used Gurobi 7.5.1 to solve LPs and integer programs. We used BARON 15.6.5 (with CPLEX 12.6 as LP solver and IPOPT as nonlinear solver) as our choice of commercial global solver, which we call from MATLAB R2015b.

5.3.2 Root node

We assess the strength of our proposed polyhedral outer approximation of light $-$ S^{SOCP} relaxation (defined in Section 5.2.1 and referred as SOCP in the tables) against the classical SDP and McCormick (Mc) relaxations. The numerical results are reported in Table 2, where SDP+Mc denotes the the intersection of SDP and Mc relaxations. Similarly, SOCP+Mc denotes the intersection of SOCP and Mc relaxations (since we are not using S^{SOCP}, this could potentially be stronger than SOCP). For easy comparison, we also have Table 3 which is similar to Table 2 except that it displays the relative deviation from SOCP rather than the actual values for bounds and running times.

Table 2: Root relaxations

	Mc		SDP		SDP+Mc		SOCP		SOCP+Mc	
Inst	Bound	Time	Bound	Time	Bound	Time	Bound	Time	Bound	Time
1	0.17771	0.07	0.17771	1.81	0.17771	35.89	0.17793	17.59	0.17793	18.42
2	0.00000	0.05	0.00000	1.70	0.00000	38.98	0.00000	20.93	0.00000	21.14
3	0.27543	0.07	0.27194	1.81	0.27543	44.02	0.28202	16.22	0.28202	49.61
4	0.10095	0.08	0.10012	2.14	0.10095	36.13	0.10101	20.71	0.10101	25.87
5	0.34766	0.05	0.34766	1.67	0.34766	31.58	0.34925	13.17	0.34925	12.88
6	0.97758	0.05	0.91629	1.80	0.97758	28.47	1.00267	11.64	1.00267	11.07
7	1.73437	0.07	1.70329	1.38	1.73437	25.29	1.74015	10.76	1.74015	11.68
8	1.99887	0.07	1.97107	1.30	1.99887	21.95	2.01260	17.53	2.01260	21.51
9	1.89400	0.05	1.89222	1.17	1.89400	22.94	1.90191	10.53	1.90191	9.32
10	2.41036	0.05	2.40658	1.16	2.41036	18.95	2.41959	10.07	2.41959	12.29
	0.99169	**0.06**	**0.97869**	**1.59**	**0.99169**	**30.42**	**0.99871**	**14.92**	**0.99871**	**19.38**

As we see, SOCP produces the best dual bounds among SDP, Mc and SDP + Mc. Also, SOCPs runs faster than SDP + Mc for all the instances. Finally, SOCP+ Mc produces no better bounds than SOCPs alone.

A strong relaxation can be obtained by partitioning the domain of some variables and writing a MILP formulation to model the union of McCormick relaxations over each piece [27, 11]. We call it McCormick Discretization and use the MILP formulation with binary expansion. We only partition the domain of variables x_i's as the number of x variables is much smaller than the number of y variables for all of our instances. In Table 4, T defines the level of discretization, meaning that the range of each variable x_i is partitioned into $2^T + 1$ uniform sub-intervals. This relaxation becomes tighter as T increases. However, the MILP that need to be solved becomes harder since the number of binary variables increases as a function of T. Thus, we give GUROBI a time limit of 10 hours, which is the amount of time given to all the branch-and-bound algorithm that we report in Section 5.3.3 below. Table 4 reports the computational results, where the asterisk signalizes that GUROBI reached the time limit

Table 3: Root relaxations

	Mc		SDP		SDP+Mc		SOCP		SOCP+Mc	
Inst	Bound	Time	Bound	Time	Bound	Time	Bound	Time	Bound	Time
1	0.999	0.004	0.999	0.103	0.999	2.040	1.000	1.000	1.000	1.047
2	1.000	0.002	1.000	0.081	1.000	1.862	1.000	1.000	1.000	1.010
3	0.977	0.004	0.964	0.112	0.977	2.714	1.000	1.000	1.000	3.059
4	0.999	0.004	0.991	0.103	0.999	1.745	1.000	1.000	1.000	1.249
5	0.995	0.004	0.995	0.127	0.995	2.398	1.000	1.000	1.000	0.978
6	0.975	0.004	0.914	0.155	0.975	2.446	1.000	1.000	1.000	0.951
7	0.997	0.007	0.979	0.128	0.997	2.350	1.000	1.000	1.000	1.086
8	0.993	0.004	0.979	0.074	0.993	1.252	1.000	1.000	1.000	1.227
9	0.996	0.005	0.995	0.111	0.996	2.179	1.000	1.000	1.000	0.885
10	0.996	0.005	0.995	0.115	0.996	1.882	1.000	1.000	1.000	1.220
	0.993	**0.004**	**0.981**	**0.111**	**0.993**	**2.087**	**1.000**	**1.000**	**1.000**	**1.271**

with the given level of discretization. If this is the case, then we report the MILP dual bound reported by the solver, which is a valid dual bound for our problem. The last column displays the best bound obtained among all the levels of discretizations reported.

Table 4: McCormick discretization: dual bounds

Inst	T=6	T=8	T=10	T=12*	T=14*	T=16*	Best
1	0.18611	0.20512	1.11852	1.85387	1.40586	0.96121	1.85387
2	0.00000	0.03133	1.05662	2.14709	1.38374	0.04654	2.14709
3	0.29443	0.33575	1.39375	2.14270	1.42642	1.42007	2.14270
4	0.10524	0.11387	1.21446	2.44853	1.63495	1.27218	2.44853
5	0.36159	0.47559	2.15416	3.40272	3.22915	2.67721	3.40272
6	1.25052	2.61325	4.16459	4.06782	3.96512	3.78165	4.16459
7	1.96682	2.17988	3.60737	4.92133	4.69632	4.47471	4.92133
8	2.48886	2.69510	3.63400	4.81890	4.48014	4.19095	4.81890
9	2.05584	2.42150	4.16064	5.54076	5.63110	5.15290	5.63110
10	2.57751	2.80795	4.07475	5.40977	5.28173	5.16376	5.40977

Clearly, McCormick discretization produces better results than $SOCP$ for these instances. Therefore, if one does not want to use branch and bound, then McCormick discretization might be the best option. However, as we see in the next section, better dual bounds can be obtained by combining SOCP with the new proposed branch-and-bound algorithm.

5.3.3 Branch-and-bound

We assess and compare the performance of the following methods:
- BB: This stands for our implementation of a branch-and-bound algorithm coded in Python.

We use GUROBI as LP solver and run IPOPT at each node to search for feasible solutions. Our algorithm uses best-bound-first as node selection and rectangular partitioning. We consider three variants that differ from each other based on the relaxation adopted in each node and in the way variables and branching points are selected:

- *SOCP-1*: Uses the polyhedral relaxation described in Section 5.2.2 with variable selection and the branching point given by Algorithm 1.
- *SOCP-2*: Uses the same relaxation of BB-SOCP-1 above. The branching variable is selected according to the gap-error-rule explained in Section 4.2. Then uses the incumbent-rule for branching point selection, whenever possible, otherwise uses the maximum-deviation-rule.
- *SOCP-3*: Same as BB-SOCP-2 except that uses bisection for branching point selection.
- *BB-Mc*: Uses McCormick relaxation with gap-error-rule as branching variable selection rule and bisection for branching point selection.

The dual bounds from our computational experiments are reported in Table 5. The stopping criteria for all the methods was a time limit of 10 hours.

Table 5: Branch-and-bound methods: dual bounds

Inst	BB-SOCP-1	BB-SOCP-2	BB-SOCP-3	BB-Mc
1	2.50744	0.18473	0.18228	0.18343
2	2.86438	0.00000	0.00000	0.00000
3	3.13078	0.29109	0.28983	0.28884
4	3.11154	0.10526	0.10246	0.10410
5	3.78958	0.35253	0.35392	0.35405
6	4.63992	1.11105	1.09537	1.15191
7	5.26603	1.99569	1.88331	1.94949
8	5.13128	2.18546	2.18193	2.28761
9	6.10860	2.17509	2.08068	2.10144
10	5.77051	2.48039	2.45158	2.47965

The best dual bound for each instance is clearly given by BB-SOCP-1, which uses our proposed relaxation and branching rule. All the standard branching rules yield significantly worse bounds.

To better understand the results of Table 5, we performed a few extra experiments in a particular instance, Instance 6, which we discuss next.

In the first experiment, we consider three variant of Algorithm 1. The first variant takes into account all type of disjunctions S_0, hyperbola, parabola or any polytope-type of objects. We name this as the *Default* setting. The second variant targets only quadrilateral-type of disjunctions, like in Figure 1. Let's name these *disconnected* objects (disconnected because both branches of the hyperbola intersect the feasible domain). In this case, only disjunctions

S_0 that are disconnected objects are considered in the loop of Line 4 in Algorithm 1. The last variant targets only hyperbola-type of disjunctions, only one branch of the hyperbola intersects the feasible domain, like in Figure 2. We name these *connected* objects. Figure 5 shows how the dual bound improves as BB-SOCP-1 iterates under each variant of Algorithm 1. Notice that when Algorithm 1 targets only disconnected objects, the quadrilaterals, the bound

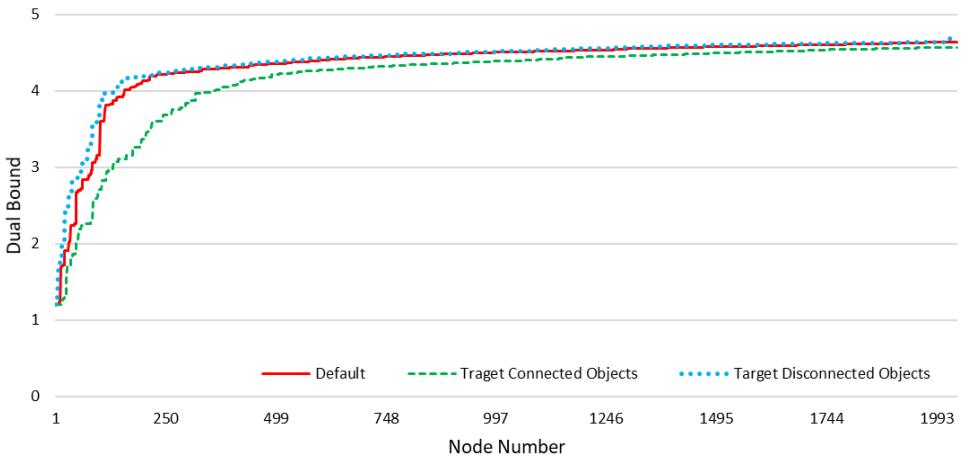

Figure 5: Dual bound improvement as BB-SOCP-1 iterates under each variant of Algorithm 1.

improvement tend to be more steep (dotted curve) than when it targets connected objects, hyperbolas. Although BB-SOCP-1 seems to perform better when Algorithm 1 targets only disconnected objects, we did not re-run all the computational experiments because after a large number of branching this performance difference is not significant.

Another plot, Figure 6, further suggests that the number of disconnected objects plays an important rule in the success of the algorithm. It shows that BB-SOCP-1 consistently eliminates disconnected objects as it goes down the tree (actually, all kind of objects are consitently eliminated). As Figure 7 shows, however, BB-SOCP-2 is not as efficient in eliminating those disconnected objects, which may explain why it doesn't perform as well as BB-SOCP-1. The behavior of BB-SOCP-3 is very similar to that of BB-SOCP-2, so the plot is not displayed here.

We also ran the following experiment to compare SOCP bound improvement versus McCormick bound improvement as a function of the number of disconnected objects. At each node branched with BB-SOCP-1 in Instance 6, we computed the following quantities: $SOCP_0$ (resp. Mc_0), the dual bound obtained with the SOCP (resp. McCormick) relaxation at the branched node; $SOCP_1$ and $SOCP_2$ (resp. Mc_1 and Mc_2), the dual bounds obtained with the SOCP (resp. McCormick) relaxation at the children of the branched node; $\Delta SOCP = SOCP_0 - \min\{SOCP_1, SOCP_2\}$ and $\Delta Mc = Mc_0 - \min\{Mc_1, Mc_2\}$; and N_0, the number of disconnected objects being branched in between x_l and x_u (see Figure 1) at the branched node. We then found that there existed a 0.58 correlation (positive) between $\Delta SOCP - \Delta Mc$ and N_0 over all branched nodes. Our interpretation of this correlation is that SOCP relaxation benefits

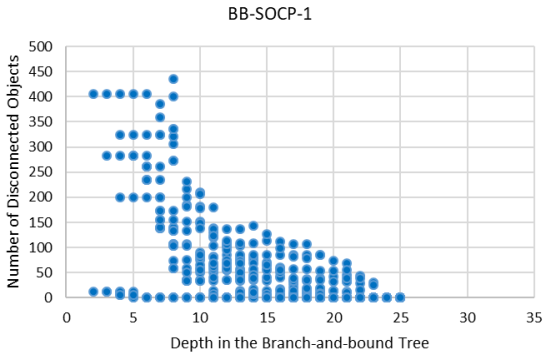

Figure 6: Number of disconnected objects encountered by BB-SOCP-1 in each node.

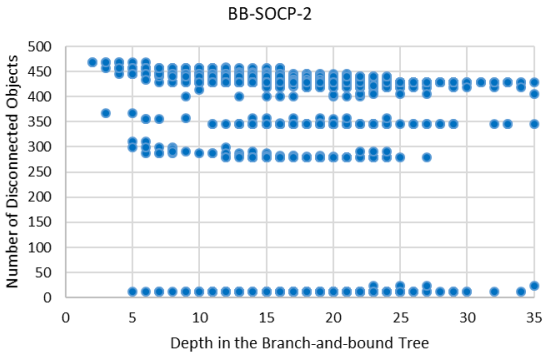

Figure 7: Number of disconnected objects encountered by BB-SOCP-2 in each node.

a lot more in the presence of disconnected objects, in comparison to the bound improvement for McCormick relaxation.

5.3.4 McCormick relaxation with BB-SOCP-1 branching rules

The computational results from Section 5.3.3, suggest that the good performance of BB-SOCP-1 is highly dependent on its branching rules, defined according to Algorithm 1. In this section we show that the branching rules of Algorithm 1 on them own are not enough to produce good dual bounds.

Consider the variant of BB-SOCP-1, referred as BB-SOCP-Mc, which uses only McCormick relaxation and the same branching rule given by Algorithm 1. Thus, at each node, we collect data from each disjunction S_0, run Algorithm 1 to select the branching variable and the branching point, but we only use the McCormick inequalities to define the relaxation.

In Table 6, we compare the performance of BB-SOCP-1 and BB-SOCP-Mc. It becomes clear that the strength of BB-SOCP-1 does not come only from the branching rules of Algorithm 1 but also from our proposed relaxation.

One reason for the discrepancy between the performance of BB-SOCP-1 and BB-SOCP-Mc is that the SOCP relaxation becomes consistently tighter than the McCormick relaxation as the algorithm goes down the branch-and-bound tree. To support our statement, we computed the dual bound from the McCormick relaxation at each node of BB-SOCP-1 on Instance 6 and compared with the dual bound from the SOCP relaxation. We found that McCormick bound was at least 25% below the SOCP bound at more than 90% of the nodes. We noticed similar behavior for other instance as well.

5.3.5 Comparison of primal bounds and duality gaps

Finally, we report in Table 7 a summary of the performance of BB-SOCP-1, McCormick Discretization, BARON and BB-Mc. Recall that the stopping criteria for all the methods was

Table 6: BB-SOCP-1 vs. McCormick relaxation with BB-SOCP-1 branching rules

	BB-SOCP-1		BB-SOCP-Mc	
Inst	Dual Bound	Gap (%)	Dual Bound	Gap (%)
1	2.50744	27.9	0.19776	94.3
2	2.86438	18.2	0.02752	99.2
3	3.13078	14.9	0.30514	91.7
4	3.11154	17.1	0.11188	97.0
5	3.78958	8.3	0.40497	90.2
6	4.63992	18.0	1.52070	73.1
7	5.26603	6.0	2.26765	59.5
8	5.13128	9.5	2.68861	52.6
9	6.10860	1.5	2.51461	59.5
10	5.77051	7.9	2.85232	54.2

a time limit of 10 hours. Also recall that primal solutions for BB-SOCP-1 and BB-Mc are obtained using IPOPT.

Table 7: Primal bounds and duality gaps

	BB-SOCP-1			Mc Disc		BARON			BB-Mc		
Inst	Dual	Primal	Gap(%)	Dual	Gap(%)	Dual	Primal	Gap(%)	Dual	Primal	Gap(%)
1	2.50744	3.47847	27.9	1.85387	46.7	0.33122	3.47887	90.5	0.18343	3.47849	94.7
2	2.86438	3.49983	18.2	2.14709	38.6	0.52447	3.49931	85.0	0.00000	3.49983	100.0
3	3.13078	3.68103	14.9	2.14270	41.8	0.47599	3.68306	87.1	0.28884	3.73308	92.3
4	3.11154	3.75223	17.1	2.44853	34.7	0.78630	3.75297	79.0	0.10410	3.75225	97.2
5	3.78958	4.13277	8.3	3.40272	17.7	0.38396	4.13541	90.7	0.35405	4.28165	91.7
6	4.63992	5.66096	18.0	4.16459	26.4	2.26566	5.66053	60.0	1.15191	5.66096	79.7
7	5.26603	5.60009	6.0	4.92133	12.1	3.07096	5.60020	45.2	1.94949	5.69318	65.8
8	5.13128	5.67022	9.5	4.81890	15.0	2.70237	5.67025	52.3	2.28761	5.67252	59.7
9	6.10860	6.20343	1.5	5.63110	9.2	3.67301	6.20346	40.8	2.10144	6.29365	66.6
10	5.77051	6.26853	7.9	5.40977	13.1	2.94060	6.22639	52.8	2.47965	6.30477	60.7

The primal bounds from all the three branch-and-bound methods are similar, suggesting that the solutions found are close to a global optimal. On the other hand, the dual bounds from BB-SOCP-1 are significantly better than the dual bounds from all the other methods, which can be seem by comparing the duality gaps. In particular, the duality gap from BB-SOCP-1 is considerably smaller than the duality gap from Mc Disc, even though we are reporting the best dual bound obtained among all the levels of discretizations $T = 6, 8, \cdots, 16$, and the primal bound we use to compute the duality gap of Mc Disc is the best primal bound from BB-SOCP-1, BARON and BB-Mc. The standard branching, i.e., the McCormick relaxation with bisection, yields the worse performance for all the instances.

6 Conclusion

6.1 Contributions

We introduced a new convexification technique for BBPs. Our procedure yields the exact convex hull of a single constraint with variables from a box domain, and we showed that this convex hull is SOCP representable. We also showed that our proposed SOCP relaxation for BBP, S^{SOCP}, is at least as tight as the SDP relaxation intersected with all the inequalities obtained from the boolean quadratic poltope (the McCormick inequalities, for example).

On the practical side, we showed how to linearly outer approximate a lighter version of S^{SOCP}, to computationally obtain much cheaper relaxations without compromising the quality of bounds in a substantial way. Moreover, moving further towards solving BBPs to global optimality, we introduced a new set of branching rules that interacts with the proposed convexification scheme and together perform much better than the bench mark commercial solver in our computational experiments.

6.2 Future research

A major practical limitation of our methodology is that it requires sparsity. Specifically, the number of variables in each bilinear constraint must be relatively small, otherwise, the resulting relaxation model cannot be solved efficiently using current software. To overcome the sparsity issue, there are ways to break down a large bilinear constraint into smaller ones involving less variables. How to do it without compromising the quality of the relaxation is an open question though.

As mentioned before, even other type of programs which are not necessarily a BBP instance can benefit from the proposed convexification scheme proposed here if there are some bilinear constraints in its definition. However, we don't know how this approach would perform in practice compared with standard relaxation techniques. An extensive computational study in this direction would be beneficial.

Another direct implication of our work is the following. Single bilinear terms of the form $w = xy$ are usually replaced with its McCormick envelope in many relaxation schemes. It's well known that when w has non-trivial bounds, i.e., tighter than those implied by the bounds of x and y, McCormick envelope does not yield the convex hull. However, this convex hull can be obtained by using our convexification technique with the addition of just a few extended variables–a description in the original space is known but fairly complicated, it's defined by a combination of multiple families of linear and nonlinear inequalities [32].

Finally, we can list two more directions for future work which deserve being explored. One is exploiting structures. Take as an example the pq-formulation of the pooling problem [17], where the domain of the variables from one set is a simplex rather than a box. Another direction is aggregating implied bilinear constraints. For example, by adding up two bilinear equations from a BBP, we obtain a new valid bilinear equation whose convexification can improve the overall relaxation. In fact, in preliminary experiments, we have seen this improvement with the instances reported here. However, at present we don't have a specific rule for combining

bilinear constraints.

Acknowledgments

The authors would like to thank Xinjun Dong in Civil and Environmental Engineering at Georgia Tech, for his assistance with preparing the structural example data. Santanu S. Dey would like to acknowledge the discussion on a preliminary version of this paper at Dagstuhl workshop # 18081, that helped improve the paper.

Funding: This work was supported by the NSF CMMI [grant number 1149400]; the NSF CMMI [grant number 1150700]; and the CNPq-Brazil [grant number 248941/2013-5].

References

[1] Al-Khayyal, F. A., and Falk, J. E. Jointly constrained biconvex programming. *Mathematics of Operations Research 8*, 2 (1983), 273–286.

[2] Belotti, P., Lee, J., Liberti, L., Margot, F., and Wächter, A. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software 24*, 4-5 (2009), 597–634.

[3] Ben-Tal, A., and Nemirovski, A. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. SIAM, 2001.

[4] Bodur, M., Del Pia, A., Dey, S. S., Molinaro, M., and Pokutta, S. Aggregation-based cutting-planes for packing and covering integer programs. *Mathematical Programming* (Sep 2017).

[5] Boland, N., Dey, S. S., Kalinowski, T., Molinaro, M., and Rigterink, F. Bounding the gap between the mccormick relaxation and the convex hull for bilinear functions. *Math. Program. 162*, 1-2 (2017), 523–535.

[6] Burer, S., Kim, S., and Kojima, M. Faster, but weaker, relaxations for quadratically constrained quadratic programs. *Computational Optimization and Applications 59*, 1 (Oct 2014), 27–45.

[7] Burer, S., and Letchford, A. N. On nonconvex quadratic programming with box constraints. *SIAM Journal on Optimization 20*, 2 (2009), 1073–1089.

[8] Castro, P. M. Tightening piecewise mccormick relaxations for bilinear problems. *Computers & Chemical Engineering 72* (2015), 300–311.

[9] Crowder, H., Johnson, E. L., and Padberg, M. Solving large-scale zero-one linear programming problems. *Operations Research 31*, 5 (1983), 803–834.

[10] Davarnia, D., Richard, J.-P. P., and Tawarmalani, M. Simultaneous convexification of bilinear functions over polytopes with application to network interdiction. *SIAM Journal on Optimization 27*, 3 (2017), 1801–1833.

[11] Dey, S. S., and Gupte, A. Analysis of milp techniques for the pooling problem. *Operations Research 63*, 2 (2015), 412–427.

[12] Dey, S. S., Molinaro, M., and Wang, Q. Analysis of sparse cutting planes for sparse milps with applications to stochastic milps. *Mathematics of Operations Research* (2017).

[13] Faria, D. C., and Bagajewicz, M. J. Novel bound contraction procedure for global optimization of bilinear minlp problems with applications to water management problems. *Computers & chemical engineering 35*, 3 (2011), 446–455.

[14] Galan, B., and Grossmann, I. E. Optimal design of distributed wastewater treatment networks. *Industrial & Engineering Chemistry Research 37*, 10 (1998), 4036–4048.

[15] Gorski, J., Pfeuffer, F., and Klamroth, K. Biconvex sets and optimization with biconvex functions: a survey and extensions. *Mathematical Methods of Operations Research 66*, 3 (2007), 373–407.

[16] Gupte, A. *Mixed integer bilinear programming with applications to the pooling problem.* PhD thesis, Georgia Institute of Technology, 2011.

[17] Gupte, A., Ahmed, S., Dey, S. S., and Cheon, M. Relaxations and discretizations for the pooling problem. *J. Global Optimization 67*, 3 (2017), 631–669.

[18] Gupte, A., Kalinowski, T., Rigterink, F., and Waterer, H. Extended formulations for convex hulls of graphs of bilinear functions. *Unpublished*.

[19] Haverly, C. A. Studies of the behavior of recursion for the pooling problem. *Acm sigmap bulletin*, 25 (1978), 19–28.

[20] Hillestad, R. J., and Jacobsen, S. E. Linear programs with an additional reverse convex constraint. *Applied Mathematics and Optimization 6*, 1 (Mar 1980), 257–269.

[21] Kim, S., and Kojima, M. Second order cone programming relaxation of nonconvex quadratic optimization problems. *Optimization methods and software 15*, 3-4 (2001), 201–224.

[22] Kocuk, B., Dey, S. S., and Sun, X. A. Matrix minor reformulation and socp-based spatial branch-and-cut method for the ac optimal power flow problem. *arXiv preprint arXiv:1703.03050* (2017).

[23] Linderoth, J. A simplicial branch-and-bound algorithm for solving quadratically constrained quadratic programs. *Mathematical Programming 103*, 2 (Jun 2005), 251–282.

[24] Luedtke, J. R., Namazifar, M., and Linderoth, J. Some results on the strength of relaxations of multilinear functions. *Math. Program. 136*, 2 (2012), 325–351.

[25] Marchand, H., and Wolsey, L. A. Aggregation and mixed integer rounding to solve mips. *Operations research 49*, 3 (2001), 363–371.

[26] Meyer, C. A., and Floudas, C. A. Convex envelopes for edge-concave functions. *Mathematical programming 103*, 2 (2005), 207–224.

[27] Meyer, C. A., and Floudas, C. A. Global optimization of a combinatorially complex generalized pooling problem. *AIChE journal 52*, 3 (2006), 1027–1037.

[28] Misener, R., and Floudas, C. A. ANTIGONE: Algorithms for coNTinuous/Integer Global Optimization of Nonlinear Equations. *J. of Global Optimization 59*, 2-3 (July 2014), 503–526.

[29] Nahapetyan, A. G. Bilinear programming: applications in the supply chain management bilinear programming: Applications in the supply chain management. In *Encyclopedia of Optimization*, C. A. Floudas and P. M. Pardalos, Eds. Springer, 2008, pp. 282–288.

[30] Nguyen, T. T., Richard, J.-P. P., and Tawarmalani, M. Deriving the convex hull of a polynomial partitioning set through lifting and projection. Tech. rep., working paper, 2013.

[31] Nguyen, T. T., Tawarmalani, M., and Richard, J.-P. P. Convexification techniques for linear complementarity constraints. In *IPCO* (2011), vol. 6655, Springer, pp. 336–348.

[32] Nguyen, T. T. T., Richard, J.-P. P., and Tawarmalani, M. Deriving the convex hull of a polynomial partitioning set through lifting and projection.

[33] Padberg, M. The boolean quadric polytope: some characteristics, facets and relatives. *Mathematical programming 45*, 1-3 (1989), 139–172.

[34] Rahman, H., and Mahajan, A. Facets of a mixed-integer bilinear covering set with bounds on variables. *arXiv preprint arXiv:1707.06712* (2017).

[35] Rikun, A. D. A convex envelope formula for multilinear functions. *Journal of Global Optimization 10*, 4 (Jun 1997), 425–437.

[36] Ryoo, H. S., and Sahinidis, N. V. A branch-and-reduce approach to global optimization. *Journal of Global Optimization 8*, 2 (Mar 1996), 107–138.

[37] Sahinidis, N. V., and Tawarmalani, M. Accelerating branch-and-bound through a modeling language construct for relaxation-specific constraints. *Journal of Global Optimization 32*, 2 (Jun 2005), 259–280.

[38] Speakman, E., and Lee, J. On branching-point selection for triple products in spatial branch-and-bound: the hull relaxation. *arXiv preprint arXiv:1706.08438* (2017).

[39] Tawarmalani, M., and Richard, J.-P. P. Decomposition techniques in convexification of inequalities. *Technical report* (2013).

[40] Tawarmalani, M., Richard, J.-P. P., and Chung, K. Strong valid inequalities for orthogonal disjunctions and bilinear covering sets. *Mathematical Programming 124*, 1 (2010), 481–512.

[41] Tawarmalani, M., Richard, J.-P. P., and Xiong, C. Explicit convex and concave envelopes through polyhedral subdivisions. *Mathematical Programming* (2013), 1–47.

[42] Tawarmalani, M., and Sahinidis, N. V. *Convexification and global optimization in continuous and mixed-integer nonlinear programming: theory, algorithms, software, and applications*, vol. 65. Springer Science & Business Media, 2002.

[43] Tuy, H. *Convex analysis and global optimization*, vol. 110. Springer, 2016.

[44] Vigerske, S., and Gleixner, A. SCIP: global optimization of mixed-integer nonlinear programs in a branch-and-cut framework. *Optimization Methods and Software 33*, 3 (2018), 563–593.

[45] Zhu, D., Dong, X., and Wang, Y. Substructure stiffness and mass updating through minimization of modal dynamic residuals. *Journal of Engineering Mechanics 142*, 5 (2016), 04016013.