

## RESEARCH ARTICLE

# A Pivoting Algorithm for Linear Programming with Linear Complementarity Constraints

Haw-ren Fang<sup>a</sup> and Sven Leyffer<sup>b\*</sup> and Todd Munson<sup>b</sup>

<sup>a</sup>*Department of Computer Science & Engineering, University of Minnesota, 200 Union St. SE., Minneapolis, MN 55455, USA;* <sup>b</sup>*Mathematics & Computer Science Division, Argonne National Laboratory, 9700 S. Cass Ave., Argonne, IL 60439, USA*

(Received 00 Month 200x; in final form 00 Month 200x)

We present a pivoting algorithm for solving linear programs with linear complementarity constraints. Our method generalizes the simplex method for linear programming to deal with complementarity conditions. We develop an anticycling scheme that can verify Bouligand stationarity. We also give an optimization-based technique to find an initial feasible vertex. Starting with a feasible vertex, our algorithm always finds a minimizer or an unbounded descent search direction in a finite number of pivoting steps.

**Keywords:** complementarity constraints; linear program with complementarity constraints; pivoting method.

**AMS Subject Classification:** 65K15, 90C33, 90C49

## 1. Introduction

In the past decade, extensive efforts have been spent on mathematical programming with equilibrium constraints (MPEC). Many of these research efforts center on constraint qualifications and stationarity conditions [40, 41, 44, 45] and algorithms based on nonlinear programming (NLP) techniques to obtain stationary points [1, 2, 8, 13, 15, 16, 19, 26, 29, 30, 35–37, 42]. Other research efforts focus on finding global solutions [25, 32], but these methods incur considerable computational costs.

A common drawback of the NLP-based approach for MPECs is that it can converge to spurious stationary points, such as C-stationary or M-stationary points, with trivial descent directions. Recently a robust method for solving MPECs was proposed [38], based on sequentially solving a linear model of an MPEC, a linear program with linear complementarity constraints (LPCC). This method converges to B-stationary points which have no trivial descent directions provided one finds B-stationary points for the LPCCs generated. Moreover, failure of the LPCC method to find a feasible point and report only B-stationary point for an infeasibility minimization problem is robustly handled by the algorithm, which will compute a B-stationary point for the MPEC infeasibility minimization problem. This algorithm motivates us to investigate new local techniques to solve LPCCs.

The global solution of LPCCs is a challenging problem, because LPCCs have non-convex feasible sets that might not even be connected. Indeed an LPCC is always transformable to a mixed-integer program, and sometimes transformable to a bilevel linear program. Several works exploit the connections in order to develop new methods. Examples

---

\*To whom correspondence should be addressed.

include cutting plane [24, 27, 28] and branch-and-bound methods [3–5, 22, 31], where the latter are a class of algorithms for finding a global minimum of a bilevel linear program. Both approaches are based on mixed-integer programming. In addition, a penalty method was also proposed by Önal [39] and analyzed by Campêlo and Scheimberg [9]. While these methods can be used to compute global solutions to LPCCs, the computational cost of applying them is large when we need only local B-stationary solutions.

Our proposed method for solving LPCCs does not guarantee a global solution, though it does provide global optimality in some of our test problems. Rather than transforming the LPCC into another type of program, we consider the LPCC itself and develop a pivoting algorithm that can handle linear complementarity constraints based on the classical simplex method for linear programming. A related local method is described in [33], but this method was developed only for nondegenerate problems. Our algorithm, however, is applicable to degenerate problem and includes an optimization-based initialization method and an anticycling scheme. In addition, certain well-established techniques for the simplex method, such as steepest-edge search [21, 43] and low-rank modification of LU factorization for active-set updates [6, 7, 17, 43], can be used in our algorithm to improve its efficiency.

The rest of this paper is organized as follows. Section 2 reviews the stationarity conditions for LPCCs. Section 3 gives a generalized pivoting algorithm for LPCC under a nondegeneracy assumption. Section 4 extends the capability of our algorithm to work under degeneracy. Section 5 gives a scheme to break pivoting cycles due to degeneracy and nonstrict complementarity conditions. This anticycling scheme can also prove B-stationarity. Our algorithm requires an initial feasible vertex to start, and Section 6 presents an optimization-based method to find such an initial vertex. Section 7 reports some numerical results.

## 2. Stationarity Conditions for LPCC

In this section we briefly review optimality conditions for LPCCs. We consider the LPCC

$$\begin{cases} \text{minimize } g^T x \\ \text{subject to } a_i^T x \geq b_i, & i = 1, \dots, m \\ 0 \leq (a_i^T x - b_i) \perp (a_{p+i}^T x - b_{p+i}) \geq 0, & i = m+1, \dots, m+p, \end{cases} \quad (1)$$

where  $x \in \mathbb{R}^n$ . The notation  $y \perp z$  means that  $y$  and  $z$  are orthogonal; that is,  $y^T z = 0$  for vectors, or simply  $yz = 0$  for real values. Without loss of generality, we have reordered the inequalities such that the last  $2p$  inequalities are in the  $p$  complementary conditions. We call inequalities  $a_i^T x \geq b_i$  *standard* constraints for  $i = 1, \dots, m$  and *complementarity* constraints for  $i = m+1, \dots, m+2p$ .

Our method is readily extended to more general forms of linear constraints, such as equality constraints, range constraints, or mixed complementarity conditions. We have chosen the format in (1) mainly to simplify the presentation.

For an index  $i$  of a complementarity constraint, we define  $c(i)$  to be the index of the constraint to which it is complementary. That is,

$$c(i) = \begin{cases} \text{NULL}, & \text{if } i \leq m; \\ i+p, & \text{if } m+1 \leq i \leq m+p; \\ i-p, & \text{if } m+p+1 \leq i \leq m+2p. \end{cases} \quad (2)$$

A point is called *linear feasible* if it satisfies the linear inequalities

$$a_i^T x \geq b_i, \quad i = 1, \dots, m+2p. \quad (3)$$

A point is called *complementary* if it satisfies the complementarity conditions

$$(a_i^T x - b_i)(a_{c(i)}^T x - b_{c(i)}) = 0, \quad i = m+1, \dots, m+p. \quad (4)$$

A point is called *feasible* if it is complementary and linear feasible, that is, satisfying all the constraints in (1).

Several stationarity concepts for optimization problems with equilibrium or complementarity constraints have been proposed. We briefly review the strong- and Bouligand-stationarity conditions for LPCCs. Other stationarity concepts such as A-, C-, L-, or M-stationarity [23, 40, 41, 44, 45] include trivial descent directions and therefore are not of interest.

We call a complementarity condition  $(a_i^T x - b_i) \perp (a_{c(i)}^T x - b_{c(i)})$  *nonstrict* at  $\hat{x}$  if  $a_i^T \hat{x} = b_i$  and  $a_{c(i)}^T \hat{x} = b_{c(i)}$  [42]. It is also sometimes called a degenerate or lower-level degenerate complementarity condition. We denote the index set of nonstrict complementarity conditions by

$$\mathcal{D}(\hat{x}) = \left\{ i : a_i^T \hat{x} = b_i \wedge a_{c(i)}^T \hat{x} = b_{c(i)}, i = m+1, \dots, m+p \right\}. \quad (5)$$

**DEFINITION 2.1** *A feasible point  $\hat{x}$  of (1) is called strongly stationary if there exist multipliers  $y_1, \dots, y_{m+2p}$ , such that*

$$\begin{cases} g - \sum_{i=1}^{m+2p} y_i a_i = 0, \\ 0 \leq (a_i^T \hat{x} - b_i) \perp y_i \geq 0, \quad \forall i \in \{1, \dots, m\}; \\ a_i^T \hat{x} > b_i \Rightarrow y_i = 0, \quad \forall i \in \{m+1, \dots, m+2p\}; \\ y_i \geq 0 \wedge y_{c(i)} \geq 0, \quad \forall i \in \mathcal{D}(\hat{x}). \end{cases} \quad (6)$$

If we relax the condition  $(a_j^T \hat{x} - b_j) \perp (a_{c(j)}^T \hat{x} - b_{c(j)})$  for  $j \in \mathcal{D}(\hat{x})$  at a feasible point  $\hat{x}$  and fix the remaining complementarity conditions as equations, similar to (2.2), then in a neighborhood of  $\hat{x}$ , the LPCC problem (1) is reduced to an LP problem. If the solution of this LP is also  $\hat{x}$ , then  $\hat{x}$  is strongly stationary. The Karush-Kuhn-Tucker (KKT) conditions of this LP at  $\hat{x}$ , which imply a minimum, are equivalent to the strong-stationarity conditions in Definition 2.1. Therefore, a strongly stationary point of LPCC (1) is a local minimizer, but there exist local minima that are not strongly stationary, see e.g. (9). We note that  $\hat{x}$  is only a local minimizer, because a different feasible point  $\bar{x}$  gives rise to a different LP. Next, we review a necessary and sufficient condition for a local minimizer of LPCC, namely B-stationarity.

For MPECs, a *Bouligand-stationary* or *B-stationary* point is a point at which no linearized feasible descent directions exist [41]. For LPCCs, an equivalent, more convenient definition is given next.

**DEFINITION 2.2** *Given a feasible point  $\hat{x}$  of (1) and a subset of nonstrict complementarity conditions  $\mathcal{P} \subseteq \mathcal{D}(\hat{x})$ , the LP piece  $LP(\hat{x}, \mathcal{P})$  is defined by tightening the nonstrict comple-*

mentarity conditions:

$$\left\{ \begin{array}{l} \text{minimize } g^T x \\ \text{subject to } a_i^T x \geq b_i, \quad \forall i \in \{1, \dots, m\}; \\ a_i^T x = b_i \text{ and } a_{c(i)}^T x \geq b_{c(i)}, \quad \forall i \in \{m+1, \dots, m+p\} \setminus \mathcal{D}(\hat{x}) \text{ and } a_i^T \hat{x} = b_i; \\ a_i^T x \geq b_i \text{ and } a_{c(i)}^T x = b_{c(i)}, \quad \forall i \in \{m+1, \dots, m+p\} \setminus \mathcal{D}(\hat{x}) \text{ and } a_{c(i)}^T \hat{x} = b_{c(i)}; \\ a_i^T x = b_i \text{ and } a_{c(i)}^T x \geq b_{c(i)}, \quad \forall i \in \mathcal{P}; \\ a_i^T x \geq b_i \text{ and } a_{c(i)}^T x = b_{c(i)}, \quad \forall i \in \mathcal{D}(\hat{x}) \setminus \mathcal{P}. \end{array} \right. \quad (7)$$

DEFINITION 2.3 We call  $\hat{x}$  a B-stationary point if and only if  $\hat{x}$  is a minimizer of all LP pieces  $LP(\hat{x}, \mathcal{P})$  for  $\mathcal{P} \subseteq \mathcal{D}(\hat{x})$ .

Remark 1 An equivalent statement in terms of multipliers is that for all  $\mathcal{P} \subseteq \mathcal{D}(\hat{x})$  there exist multipliers  $y_1, \dots, y_{m+2p}$ , such that

$$\left\{ \begin{array}{l} g - \sum_{i=1}^{m+2p} y_i a_i = 0, \\ 0 \leq (a_i^T \hat{x} - b_i) \perp y_i \geq 0, \quad \forall i \in \{1, \dots, m\}; \\ a_i^T \hat{x} > b_i \Rightarrow y_i = 0, \quad \forall i \in \{m+1, \dots, m+2p\} \setminus \mathcal{D}(\hat{x}); \\ y_{c(i)} \geq 0, \quad \forall i \in \mathcal{P}; \\ y_i \geq 0, \quad \forall i \in \mathcal{D}(\hat{x}) \setminus \mathcal{P} \end{array} \right. \quad (8)$$

holds.

In general, strong stationarity implies B-stationarity, but not vice versa. However, when all complementarity constraints are strict, then strong stationarity and B-stationarity are equivalent.

Scheel and Scholtes [41], page 8 give an example where a vertex is B-stationary but not strongly stationary:

$$\left\{ \begin{array}{l} \text{minimize } x_1 + x_2 - x_3 \\ \text{subject to } 4x_1 - x_3 \geq 0, \quad \text{indexed by 1;} \\ \quad \quad \quad 4x_2 - x_3 \geq 0, \quad \text{indexed by 2;} \\ \quad \quad \quad 0 \leq x_1 \perp x_2 \geq 0, \text{ indexed by 3 and 4.} \end{array} \right. \quad (9)$$

The only feasible vertex of (9) is  $(0, 0, 0)$ . By Definition 2.1, strong stationarity requires that there exist nonnegative multipliers  $y_1, y_2, y_3, y_4$  satisfying

$$\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 4 & 0 & 1 & 0 \\ 0 & 4 & 0 & 1 \\ -1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}. \quad (10)$$

Adding  $4y_1 + y_3 = 1$  and  $4y_2 + y_4 = 1$  minus four times  $y_1 + y_2 = 1$ , we obtain  $y_3 + y_4 = -2$ , which contradicts  $y_3 \geq 0$  and  $y_4 \geq 0$ . Thus,  $(0, 0, 0)$  is not strongly stationary.

To prove B-stationarity, we observe that the two LP pieces of (9) correspond to  $x_1 = 0 \leq x_2$  and  $x_1 \geq 0 = x_2$ . Using (10), we obtain the multipliers  $(\frac{3}{4}, \frac{1}{4}, -2, 0)$  and  $(\frac{1}{4}, \frac{3}{4}, 0, -2)$  for the two LP pieces. Thus,  $(0, 0, 0)$  is B-stationary.

### 3. Pivoting under Nondegeneracy

Our algorithm generalizes the active-set method for LP to LPCC. The algorithm starts at a feasible vertex  $\hat{x}$  and moves from one vertex to another along a feasible edge to reduce  $g^T x$ . For ease of presentation, we make the following two assumptions.

- (1) There are exactly  $n$  linearly independent active constraints at every vertex.
- (2) An initial feasible vertex is given and associated with  $n$  linearly independent active constraints.

The first assumption is a general nondegeneracy assumption, and we show in Sections 4 and 5 how to remove it, by extending the working set and developing a suitable anticycling scheme for LPCC. The second assumption can be removed by a two-phase process, which we describe in Section 6.

A vertex  $\hat{x}$  of LPCC (1) is determined by a working set of  $n$  active linearly independent constraints, whose indices form a set denoted by  $\mathcal{W}$ . Under the nondegeneracy assumption, all active constraints are in the working set  $\mathcal{W}$ . To satisfy the complementarity condition in (1), we need the following additional condition:

$$\{i, c(i)\} \cap \mathcal{W} \neq \emptyset, \quad \forall i \in \{m+1, \dots, m+p\}, \quad (11)$$

where  $c(i)$ , defined in (2), is the index of the other complementarity constraint of constraint  $i$ . Condition (11) ensures that at least one constraint in each complementarity condition is active. For ease of presentation, we partition  $\mathcal{W}$  into  $\mathcal{W}_0$  and  $\mathcal{W}_1$ :

$$\mathcal{W}_0 = \mathcal{W} \cap \{1, \dots, m\}, \quad \mathcal{W}_1 = \mathcal{W} \cap \{m+1, \dots, m+2p\}, \quad (12)$$

where  $\mathcal{W}_0$  and  $\mathcal{W}_1$  contain the standard and complementarity constraints from  $\mathcal{W}$ , respectively.

We note that we have assumed nondegeneracy but not strictness of complementarity conditions. In other words, two constraints in a complementarity condition can be in the working set at the same time:  $\{j, c(j)\} \subseteq \mathcal{W}$  for some  $j \in \{m+1, \dots, m+p\}$ .

Aggregating all constraints in the working set  $\mathcal{W}$ , we obtain a linear system  $A^T x = b$ , where  $A = [a_j]_{j \in \mathcal{W}}$  and  $b = [b_j]_{j \in \mathcal{W}}$ . The Lagrangian of (1) is

$$L(x, y) = g^T x - y^T (A^T x - b).$$

The vertex  $\hat{x}$  determined by the working set  $\mathcal{W}$  is  $A^{-T} b$ . Setting  $\partial L / \partial x = 0$ , we obtain the multipliers  $\hat{y} \equiv [\hat{y}_j]_{j \in \mathcal{W}} := A^{-1} g$ . Moving from one vertex to another along a feasible edge implies replacing one entry in the working set  $\mathcal{W}$  by another, and  $A = [a_j]_{j \in \mathcal{W}}$  and  $b = [b_j]_{j \in \mathcal{W}}$  will be updated accordingly. In practice, we do not form  $A^{-1}$  but work with numerically stable LU factors. Since only one column of  $A$  is changed at each pivoting step, we can apply a rank-one update to the LU factors for computational efficiency [6, 7, 17, 43].

We denote  $A^{-T} = [s_j]_{j \in \mathcal{W}}$ . Moving from the vertex  $\hat{x} = A^{-T} b$  along the direction  $s_j$  increases  $a_j^T x_j$ , so the constraint  $a_j^T x_j > b_j$  becomes inactive, while the other equations in  $A^T x = b$  remain satisfied. The direction  $s_j$  is associated with the edge formed by  $A^T x = b$  after removing  $a_j^T x = b_j$ . The rate of change of the objective function when moving from  $\hat{x}$  to  $\hat{x} + s_j$  is given by the multiplier  $\hat{y}_j = s_j^T g$ . Therefore,  $s_j$  is a descent direction if and only if  $\hat{y}_j < 0$ .

Now we discuss whether moving along  $s_j$  from a feasible vertex  $\hat{x}$  will violate the complementarity conditions (4). We have assumed that at least one constraint in each complementarity condition is in the working set. Therefore, if constraint  $j \in \mathcal{W}_0$  (i.e.,

standard) the direction  $s_j$  does not violate (4). Otherwise,  $j \in \mathcal{W}_1$  is complementary. Under the nondegeneracy assumption, the direction  $s_j$  does not violate (4) if and only if constraint  $c(j)$  is in the working set. Thus, we may choose to drop any constraint from the following set of eligible constraints:

$$\{i : \hat{y}_i < 0 \wedge (i \in \mathcal{W}_0 \vee (i \in \mathcal{W}_1 \wedge c(i) \in \mathcal{W}_1))\}.$$

In our implementation we choose to drop the constraint with the most negative multiplier. In other words,

$$\hat{y}_q = \min\{0, \hat{y}_i : i \in \mathcal{W}_0 \vee (i \in \mathcal{W}_1 \wedge c(i) \in \mathcal{W}_1)\},$$

where  $q$  is the constraint index of the minimizer. As Lemma 3.1 will show, if  $\hat{y}_q = 0$ , then vertex  $\hat{x}$  is strongly stationary. Otherwise,  $s_q$  is the descent search direction associated with the leaving constraint  $q$ .

We also need to maintain linear feasibility (3). When moving along the direction  $s_q$ , an inactive constraint  $a_j^T x \geq b_j$  can become active only if  $a_j^T s_q$  less than 0. The maximum step length  $\alpha$  that maintains feasibility of the inactive constraint  $a_j^T x \geq b_j$  with  $a_j^T s_q < 0$  is then  $\alpha = (b_j - a_j^T \hat{x}) / a_j^T s_q \geq 0$ . Thus, the largest step length that we can take is the minimum such  $\alpha$  over all inactive constraints, namely,

$$\min \left\{ \frac{b_j - a_j^T \hat{x}}{a_j^T s_q} : a_j^T s_q < 0, j \notin \mathcal{W} \right\}, \quad (13)$$

where  $r$  is the index minimizer indicating the entering constraint. We call (13) the *ratio test*.

If  $a_j^T s_q \geq 0$  for all inequalities  $j$  not in the working set, there exists no stopping constraint, and the direction  $s_q$  is unbounded. In this case we let  $\hat{\alpha}_r$  be  $\infty$ , and we conclude that the LPCC is unbounded.

When the leaving constraint  $q$  and entering constraint  $r$  are determined, we remove constraint  $q$  from the working set, add constraint  $r$  (i.e.,  $\mathcal{W} := \mathcal{W} \cup \{r\} \setminus \{q\}$ ), and update  $A = [a_j]_{j \in \mathcal{W}}$  and  $b = [b_j]_{j \in \mathcal{W}}$ . This discussion is summarized in Algorithm 1, which follows the same principle of the active-set algorithm for MPECs by Júdice et al. [33] under the nondegeneracy assumption.

Consider Algorithm 1. Under the assumption of nondegeneracy, we can always move downhill with  $\hat{\alpha}_r > 0$  to another vertex, until no decrease is possible and a solution is found, or until the LPCC is determined unbounded. Since only a finite number of vertices exist, Algorithm 1 always terminates in a finite number of steps.

**LEMMA 3.1** *If Algorithm 1 terminates with  $\hat{y}_q = 0$ , then the final vertex  $\hat{x}$  is strongly stationary.*

*Proof* All constraints not in the working set  $\mathcal{W}$  are associated with zero multipliers. Since  $\hat{y}_q = 0$ , it follows that  $\min\{\hat{y}_i : i \in \mathcal{W}_0\} \geq 0$ , which implies that the first three conditions of (6) are satisfied. For the last condition of (6), we observe, that  $\hat{y}_q = 0$  implies that  $\min\{\hat{y}_i : i \in \mathcal{W}_1 \wedge c(i) \in \mathcal{W}_1\} \geq 0$  which guarantees that the inequalities in the nonstrict complementarity conditions have nonnegative multipliers, so all conditions for strong stationarity in Definition 2.1 are met. ■

**LEMMA 3.2** *A nondegenerate B-stationary point of an LPCC is also strongly stationary.*

*Proof* See [41, Theorem 4]. ■

```

1: // Given vertex  $\hat{x}$  associated with a working set  $\mathcal{W}$  satisfying (11).
2: Form  $A := [a_j]_{j \in \mathcal{W}}$  and  $b := [b_j]_{j \in \mathcal{W}}$ .
3: Compute current vertex  $\hat{x} := A^{-T}b$ .
4: repeat
5:   Compute multipliers  $\hat{y} \equiv [\hat{y}_i]_{i \in \mathcal{W}} := A^{-1}g$ .
6:   Compute  $\hat{y}_q := \min \{0, \hat{y}_i : i \in \mathcal{W}_0 \vee (i \in \mathcal{W}_1 \wedge c(i) \in \mathcal{W}_1)\}$ .
7:   // The index  $q$  indicates the constraint to leave the working set.
8:   if  $\hat{y}_q = 0$  then
9:     return:  $\hat{x}$  is a strongly stationary point.
10:  else
11:    Compute search direction  $s_q$  as the column of  $A^{-T}$  corresponding to  $\hat{y}_q$ .
12:    Perform the ratio test:  $\hat{\alpha}_r := \min \left\{ \frac{b_j - a_j^T \hat{x}}{a_j^T s_q}, \infty \right\}$ .
13:    // The index  $r$  indicates the constraint to enter the working set.
14:    if  $\hat{\alpha}_r = \infty$  then
15:      return: the LPCC is unbounded.
16:    else
17:      Update  $\mathcal{W} := \mathcal{W} \cup \{r\} \setminus \{q\}$ ,  $A := [a_j]_{j \in \mathcal{W}}$ , and  $b := [b_j]_{j \in \mathcal{W}}$ .
18:      Update the vertex  $\hat{x} = \hat{x} + \hat{\alpha}_r s_q$ .
19:    end if
20:  end if
21: until  $\hat{x}$  is strongly stationary or  $\hat{\alpha}_r = \infty$ .

```

**Algorithm 1:** A generalized pivoting algorithm for LPCC (1).

Now we illustrate the application of Algorithm 1 using the following example:

$$\left\{ \begin{array}{ll}
\underset{x_1, x_2, x_3, x_4, x_5}{\text{minimize}} & 4x_1 - 2x_2 + x_3 - x_5 \\
\text{subject to} & x_1 \geq 0, \quad x_2 \geq 0, \quad \text{indexed by 1,2;} \\
& x_1 + 2x_4 \geq 2, \quad \text{indexed by 3;} \\
& x_3 - x_4 - x_5 \geq -2, \quad \text{indexed by 4;} \\
& 0 \leq x_3 \perp x_1 - x_2 + x_3 + 1 \geq 0, \quad \text{indexed by 5 and 8;} \\
& 0 \leq x_4 \perp x_1 - x_3 + 2 \geq 0, \quad \text{indexed by 6 and 9;} \\
& 0 \leq x_5 \perp x_3 - x_4 + 1 \geq 0, \quad \text{indexed by 7 and 10.}
\end{array} \right. \quad (14)$$

*The first pivoting step:* As will be seen in Section 6, our initialization scheme finds a feasible vertex  $\hat{x} = (2, 0, 0, 0, 0)$  associated with the working set  $\mathcal{W} = \{2, 3, 5, 6, 7\}$ . The objective is  $g^T \hat{x} = 8$ . The multipliers  $\hat{y} \equiv [\hat{y}_j]_{j \in \mathcal{W}} = A^{-1}g$  are

$$\begin{bmatrix} \hat{y}_2 \\ \hat{y}_3 \\ \hat{y}_5 \\ \hat{y}_6 \\ \hat{y}_7 \end{bmatrix} = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 2 & 1 \\ & & & & 1 \end{bmatrix}^{-1} \begin{bmatrix} 4 \\ -2 \\ 1 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} -2 \\ 4 \\ 1 \\ -8 \\ -1 \end{bmatrix}.$$

The most negative multiplier is  $\hat{y}_6 = -8$ , associated with constraint 6, which is, however, complementary. The other complement, indexed by 9, is inactive and not in the working set. Thus, we cannot remove this constraint from the working set. The second most negative multiplier is  $\hat{y}_2 = -2$ , and constraint  $q = 2$  will leave the basis. The ratio test shows that  $a_8^T s_q < 0$ , and constraint  $r = 8$  will enter the basis.

**The second pivoting step:** Now the vertex associated with the working set  $\mathcal{W} = \{3, 5, 6, 7, 8\}$  is  $\hat{x} = (2, 3, 0, 0, 0)$ . The objective is  $g^T \hat{x} = 2$ , and the multipliers are  $(\hat{y}_3, \hat{y}_5, \hat{y}_6, \hat{y}_7, \hat{y}_8) = (2, -1, -4, -1, 2)$ . There are three negative multipliers:  $\hat{y}_5 = -1$ ,  $\hat{y}_6 = -4$ , and  $\hat{y}_7 = -1$ . The complementarity constraints 6 and 7 cannot leave the working set, since their other complements are inactive and not in the working set. The leaving constraint is  $q = 5$ . The ratio test (13) determines the entering constraint  $r = 9$ .

**The third pivoting step:** We are now at vertex  $\hat{x} = (2, 7, 4, 0, 0)$ , determined by the working set  $\mathcal{W} = \{3, 6, 7, 8, 9\}$ . The objective is  $g^T \hat{x} = -2$ . The multipliers are  $(\hat{y}_3, \hat{y}_6, \hat{y}_7, \hat{y}_8, \hat{y}_9) = (1, -2, -1, 2, 1)$ . There are two negative multipliers,  $\hat{y}_6 = -2$  and  $\hat{y}_7 = -1$ . As before, the complementarity constraint 7 cannot leave the working set  $\mathcal{W}$ . The leaving constraint is  $q = 6$ . The ratio test (13) determines the entering constraint  $r = 1$ .

**The result:** Now the multipliers associated with working set  $\mathcal{W} = \{1, 3, 7, 8, 9\}$  are  $(\hat{y}_1, \hat{y}_3, \hat{y}_7, \hat{y}_8, \hat{y}_9) = (1, 0, -1, 2, 1)$ . The only negative multiplier  $\hat{y}_7 = -1$  is associated with the complementarity constraint 7, which cannot leave the working set since the other complement, indexed by 10, is inactive and not in working set  $\mathcal{W}$ . Therefore, Algorithm 1 terminates at  $\hat{x} = (0, 3, 2, 1, 0)$ , which is strongly stationary. The final objective is  $g^T \hat{x} = -4$ .

#### 4. Pivoting under Degeneracy

In this section we extend Algorithm 1 by allowing degenerate vertices. The complementarity constraints that can leave the working set are

$$C = \{i : i \in \mathcal{W}_1 \wedge c(i) \in \mathcal{W}_1\}.$$

In other words, under the nondegeneracy assumption, complementarity constraint  $i$  can leave the working set without violating (4) if constraint  $c(i)$  remains in the working set. However, at a degenerate vertex  $\hat{x}$ , we must also take complementarity constraint  $i$  into account, if constraint  $c(i)$  is active but not in the working set. Otherwise, Lemma 3.1 is no longer valid. Therefore, we extend the candidate set  $C$  by including all active complementarity constraints to

$$\bar{C} = \{i : i \in \mathcal{W}_1 \wedge a_{c(i)}^T \hat{x} = b_{c(i)}\}.$$

Another issue arises if complementarity constraint  $q \in \bar{C} \setminus C$  leaves and constraint  $c(q)$  is not in the working set  $\mathcal{W}$ , because in this case,  $(a_q^T x - b_q) \perp (a_{c(q)}^T x - b_{c(q)})$  may be violated after the pivot. A naive solution is that whenever  $q \in \bar{C} \setminus C$  leaves the working set, we immediately add  $c(q)$  into the working set, that is, update  $\mathcal{W} := \mathcal{W} \cup \{q\} \setminus \{c(q)\}$ . The pitfall is that the resulting working matrix  $A = [a_i]_{i \in \mathcal{W}}$  may become singular.

For example, consider the LPCC

$$\left\{ \begin{array}{l} \text{minimize } -x_1 \\ \quad \quad \quad x_1, x_2, x_3 \\ \text{subject to } x_1 - x_2 + x_3 \geq 0, \text{ indexed by 1;} \\ \quad \quad \quad x_1 + x_2 + x_3 \geq 0, \text{ indexed by 2;} \\ \quad \quad \quad -x_1 \geq -1, \quad \quad \quad \text{indexed by 3;} \\ \quad \quad \quad 0 \leq x_1 \perp x_2 \geq 0, \text{ indexed by 4 and 5.} \end{array} \right. \quad (15)$$

Say we are at vertex  $(\hat{x}_1, \hat{x}_2, \hat{x}_3) = (0, 0, 0)$ , associated with the initial working set  $\mathcal{W} = \{1, 2, 4\}$ . The associated working matrix  $A = [a_j]_{j \in \mathcal{W}}$ , the objective normal  $g$ , and the

multipliers  $\hat{y} := A^{-1}g$  are

$$A = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}, \quad g = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}, \quad \hat{y} = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}. \quad (16)$$

The only negative multiplier is  $\hat{y}_3 = -1$ , associated with the leaving constraint  $x_1 \geq 0$ . The other complementarity constraint  $x_2 \geq 0$  is active but not in the working set  $\mathcal{W}$ . To maintain  $0 \leq x_1 \perp x_2 \geq 0$ , we add constraint 5 into the working set, giving  $\mathcal{W} = \{1, 2, 5\}$ .

The updated working matrix  $A = \begin{bmatrix} 1 & 1 & 0 \\ -1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$  is singular.

To deal with degeneracy in LPCC, we introduce the concept of *extended* working set:

$$\bar{\mathcal{W}} = \mathcal{W} \cup \mathcal{E}, \quad \mathcal{W} \cap \mathcal{E} = \emptyset,$$

where the set  $\mathcal{E}$  is an extension. All constraints in  $\bar{\mathcal{W}}$  are active at the current vertex  $\hat{x}$ . While we still maintain the working set  $\mathcal{W}$  consisting of  $n$  linearly independent active constraints, the extension  $\mathcal{E}$  contains the complementarity constraints that should be kept active to satisfy (4). Thus, condition (11) is relaxed to

$$\{i, c(i)\} \cap \bar{\mathcal{W}} \neq \emptyset \quad \text{for } i = m+1, \dots, m+p. \quad (17)$$

The high level description of the revised algorithm is as follows. If the leaving constraint  $q$  is complementary and  $c(q)$  is not in the extended working set  $\bar{\mathcal{W}} = \mathcal{W} \cup \mathcal{E}$ , then we add  $c(q)$  to  $\mathcal{E}$ . To determine the entering constraint, if a positive step length would cause any constraint  $r \in \mathcal{E}$  to no longer be active, we move  $r$  from  $\mathcal{E}$  to  $\mathcal{W}$  right away. Otherwise, we proceed with the usual ratio test (13) to determine the entering constraint  $r$ , and remove  $c(r)$  from  $\mathcal{E}$  if  $c(r) \in \mathcal{E}$ . More details are given in Algorithm 2.

Five remarks on Algorithm 2 must be made:

- (1) Lemma 3.1 remains valid. If the algorithm terminates with  $\hat{y}_q = 0$ , then the final vertex  $\hat{x}$  is strongly stationary.
- (2) At the end of each pivoting step, we keep all constraints in  $\bar{\mathcal{W}}$  active and maintain (17) so that at least one constraint in each complementarity condition is in  $\bar{\mathcal{W}}$ . Therefore, the vertices visited are always feasible.  
Note that, in line 14 of Algorithm 2,  $a_r^T s_q \neq 0$  for some  $r \in \mathcal{E}$  means that moving along  $s_q$  will make constraint  $r$  inactive. So we move  $r$  from  $\mathcal{E}$  into  $\mathcal{W}$  immediately, in which case the resulting vertex  $\hat{x}$  is unchanged, and therefore the other constraints in  $\mathcal{E}$ , if any, remain active. Moreover, we need to keep  $\mathcal{W}$  of size  $n$ . Therefore, if there are multiple choices of  $r \in \mathcal{E}$  with  $a_r^T s_q \neq 0$  in line 14, we move only one of them to  $\mathcal{W}$ .
- (3) In addition,  $A = [a_j]_{j \in \mathcal{W}}$  is always guaranteed to be nonsingular. The reason is that no matter which entering constraint  $r$  is chosen in line 14 of Algorithm 2 or in the ratio test in line 17, we have  $a_r^T s_q \neq 0$ .
- (4) Here and throughout this paper, our extension set  $\mathcal{E}$  contains only complementarity constraints of (1), namely,  $\forall i \in \mathcal{E}, i > m$ . Under a nondegeneracy assumption,  $\mathcal{E} = \emptyset$  and Algorithm 2 coincides with Algorithm 1.
- (5) We may impose the following condition,

$$\forall j \in \mathcal{E}, \quad c(j) \notin \bar{\mathcal{W}}, \quad (18)$$

to exclude unnecessary complementarity constraints from  $\mathcal{E}$  and therefore po-

```

1: // Given vertex  $\hat{x}$  associated with  $\bar{\mathcal{W}} = \mathcal{W} \cup \mathcal{E}$  satisfying (17).
2: // All constraints in  $\bar{\mathcal{W}}$  are active at  $\hat{x}$ ;  $\mathcal{W}$  contains  $n$  linearly independent constraints.
3:  $\mathcal{A}_1 := \emptyset$ ;  $\mathcal{A}_2 := \emptyset$ ; form  $A := [a_j]_{j \in \mathcal{W}}$  and  $b := [b_j]_{j \in \mathcal{W}}$ .
4: Compute current vertex  $\hat{x} := A^{-T}b$ .
5: repeat
6:   Compute multipliers  $\hat{y} \equiv [\hat{y}_i]_{i \in \mathcal{W}} := A^{-1}g$ .
7:   Compute  $\hat{y}_q := \min \{0, \hat{y}_i : i \in \mathcal{W}_0 \vee (i \in \mathcal{W}_1 \wedge c(i) \in \mathcal{W}_1)\}$ .
8:   // The index  $q$  indicates the constraint to quit the working set.
9:   if  $\hat{y}_q = 0$  then
10:    return:  $\hat{x}$  is strongly stationary.
11:   else
12:     if  $q \in \mathcal{W}_1$  and  $c(q) \notin \bar{\mathcal{W}}$  then
13:        $\mathcal{E} := \mathcal{E} \cup \{c(q)\}$ .
14:     end if
15:     Compute search direction  $s_q$  as the column of  $A^{-T}$  corresponding to  $\hat{y}_q$ .
16:     if  $\exists r \in \mathcal{E}$  such that  $a_r^T s_q \neq 0$  then
17:        $\mathcal{E} := \mathcal{E} \setminus \{r\}$ 
18:     else // The index  $r$  indicates the constraint to enter the working set.
19:       Ratio test:  $\hat{\alpha}_r := \min_{\substack{j \in \mathcal{W} \\ a_j^T s_q < 0}} \left\{ \frac{b_j - a_j^T \hat{x}}{a_j^T s_q}, \infty \right\}$ .
20:       if  $\hat{\alpha}_r = \infty$  then
21:         return: the LPCC is unbounded.
22:       end if
23:     end if
24:     Update the working set  $\mathcal{W} := \mathcal{W} \cup \{r\} \setminus \{q\}$ .
25:     if  $c(r) \in \mathcal{E}$  then
26:       Remove  $c(r)$ ;  $\mathcal{E} := \mathcal{E} \setminus \{c(r)\}$ .
27:     end if
28:      $(\text{cycle}, \mathcal{A}_1, \mathcal{A}_2) = \text{DETECTCYCLE}(q, r, \hat{\alpha}_r, \mathcal{A}_1, \mathcal{A}_2)$ ; // see Algorithm 3.
29:     if  $\text{cycle} = \text{true}$  then
30:        $(\text{status}, \mathcal{W}, \mathcal{E}, \hat{x}) = \text{ANTICYCLE}(\mathcal{W}, \mathcal{E}, \hat{x})$ ; // see Algorithm 4.
31:       if  $\text{status} = \text{B\_stationary}$  then
32:         return:  $\hat{x}$  is B-stationary.
33:       else if  $\text{status} = \text{unbounded}$  then
34:         return: the LPCC is unbounded.
35:       end if
36:        $\text{status} = \text{cycle\_breaks}$ 
37:     else
38:       Update the vertex  $\hat{x} := \hat{x} + \hat{\alpha}_r s_q$ .
39:     end if
40:     Update  $A := [a_j]_{j \in \mathcal{W}}$  and  $b := [b_j]_{j \in \mathcal{W}}$ .
41:   end if
42: until  $\hat{x}$  is strongly stationary or B-stationary, or  $\hat{\alpha}_r = \infty$ .

```

**Algorithm 2:** A revised pivoting algorithm for LPCC (1).

tentially reduce the number of pivoting steps. If the initial extended working set satisfies (18), then (18) remains satisfied in line 23.

- (6) If there exist multiple indices  $r \in \mathcal{E}$  such that  $a_r^T s_q \neq 0$  in Line 16, then we can choose any one of them to be removed from  $\mathcal{E}$ .

Now we apply Algorithm 2 to the LPCC (15). Let the initial working set  $\mathcal{W}$  be  $\{1, 2, 4\}$

with zero extension  $\mathcal{E} = \emptyset$  that determines the initial vertex  $(\hat{x}_1, \hat{x}_2, \hat{x}_3) = (0, 0, 0)$ . Condition (17) is satisfied. From (16) we conclude that the leaving constraint  $q$  is 4, associated with the only negative multiplier  $\hat{y}_q = -1$ . The descent direction  $s_q$  is  $(1, 0, -1)$ , the last column of  $A^{-T}$ . Because the other complementarity constraint  $c(q) = 5 \notin \mathcal{W} \cup \mathcal{E}$ , we add it to  $\mathcal{E}$  and have  $\mathcal{E} = \{5\}$ . At the test in line 14, constraint 5 from  $\mathcal{E}$  has gradient  $a_5 = (0, 1, 0)$ . Since  $a_5^T s_q = 0$ , it cannot be the entering constraint. The entering constraint is  $-x_1 \geq -1$ , determined by ratio test in line 17. The updated working set is  $\mathcal{W} = \{1, 2, 3\}$  and the vertex is  $\hat{x} = (1, 0, -1)$ . The multipliers  $(\hat{y}_1, \hat{y}_2, \hat{y}_3) = (0, 0, 1)$  are all nonnegative. As a result, the vertex  $\hat{x} = (1, 0, -1)$  is strongly stationary.

Another problem caused by degeneracy is the fact that pivoting may cycle. We will address this issue in the next section.

## 5. Anticycling for LPCC

At a degenerate vertex  $\hat{x}$ , the step length  $\hat{\alpha}_r$  can be zero, and we call such a pivoting step *degenerate*. If the degenerate pivoting steps form a cycle, it loops indefinitely. In particular, Algorithm 2, without anticycling, can terminate only at a strongly stationary point or by finding an unbounded search direction. Therefore, at a B-stationary point that is not strongly stationary, it must loop forever. For example, if we apply Algorithm 2 to the program (9) whose only vertex is B-stationary but not strongly stationary, then the step length is always zero, resulting in an infinite loop.

At each pivoting step in our algorithm, we may have multiple choices of leaving constraint and entering constraint. In LP, we can use Bland's rule [10, Theorem 3.3]; [20, Theorem 8.3.1] to resolve degeneracy. Unfortunately, simply applying Bland's rule to Algorithm 2 for LPCCs can still result in a cycle. Even worse, without anticycling, Algorithm 2 cannot determine any B-stationary point that is not strongly stationary. We illustrate the failure of Bland's rule with the following example.

$$\left\{ \begin{array}{ll} \underset{x_1, x_2, x_3}{\text{minimize}} & x_1 + x_2 + x_3 + x_4 - x_5 - x_6 \\ \text{subject to} & 4x_1 - x_5 \geq 0, \quad \text{indexed by 1;} \\ & 4x_2 - x_5 \geq 0, \quad \text{indexed by 2;} \\ & 4x_3 - x_6 \geq 0, \quad \text{indexed by 3;} \\ & 4x_4 - x_6 \geq 0, \quad \text{indexed by 4;} \\ & 0 \leq x_1 \perp x_2 \geq 0, \quad \text{indexed by 5 and 6;} \\ & 0 \leq x_3 \perp x_4 \geq 0, \quad \text{indexed by 7 and 8.} \end{array} \right. \quad (19)$$

The only feasible vertex is  $(0, 0, 0, 0, 0, 0)$ . Let the initial working set  $\mathcal{W}$  be  $\{1, 2, 3, 4, 5, 7\}$ . The multipliers  $(\hat{y}_1, \hat{y}_2, \hat{y}_3, \hat{y}_4, \hat{y}_5, \hat{y}_7)$  are  $(\frac{3}{4}, \frac{1}{4}, \frac{3}{4}, \frac{1}{4}, -2, -2)$ . By Bland's rule, the leaving constraint is  $x_1 \geq 0$ . The entering constraint is  $x_2 \geq 0$ , resulting in working set  $\mathcal{W} = \{1, 2, 3, 4, 6, 7\}$ . The multipliers  $(\hat{y}_1, \hat{y}_2, \hat{y}_3, \hat{y}_4, \hat{y}_6, \hat{y}_7)$  are  $(\frac{1}{4}, \frac{3}{4}, \frac{3}{4}, \frac{1}{4}, -2, -2)$ . Now the leaving and entering constraints are  $x_2 \geq 0$  and  $x_1 \geq 0$ . That forms cycling.

To resolve degeneracy for LPCCs, we require two routines: a cycle detection routine and an anticycling routine. We adopt a cycle detection from Chvátal [10]. We keep an array of leaving constraints  $\mathcal{A}_1$  and another array of entering constraints  $\mathcal{A}_2$ . When the last  $k$  entries of  $\mathcal{A}_1$  and  $\mathcal{A}_2$  match each other for some  $k$ , cycling is detected. Note that whenever a step length is positive (i.e.,  $\hat{\alpha}_r > 0$ ), a cycle breaks, and we reset  $\mathcal{A}_1$  and  $\mathcal{A}_2$  to be empty. The description is stated in pseudo-code in Algorithm 3.

Definition 2.3 shows how to determine whether a given vertex  $\hat{x}$  is B-stationary when nonstrict complementarity conditions are present. When a cycling at  $\hat{x}$  is detected, we consider the LP pieces  $\text{LP}(\hat{x}, \mathcal{P})$  for all possible  $\mathcal{P} \subseteq \mathcal{D}(\hat{x})$ . We apply an anticycling rule, such as Bland's least index rule, to each  $\text{LP}(\hat{x}, \mathcal{P})$ . Then it follows that  $\hat{x}$  is B-stationary

```

1: function [cycle,  $\mathcal{A}_1, \mathcal{A}_2$ ] = CYCLEDETECT( $q, r, \hat{\alpha}_r, \mathcal{A}_1, \mathcal{A}_2$ )
2:   //  $\mathcal{A}_1$  and  $\mathcal{A}_2$  are arrays containing the recent leaving and entering constraints.
3:   //  $q$  and  $r$  are the leaving and entering constraints;  $\hat{\alpha}_r$  is the step length.
4:   if  $\hat{\alpha}_r > 0$  then
5:     // The objective of (1) is reduced, so previous steps cannot trigger a cycling.
6:      $\mathcal{A}_1 := \emptyset$ ;  $\mathcal{A}_2 := \emptyset$ 
7:     cycle := false
8:   else
9:     Update  $l := \text{size}(\mathcal{A}_1)$ ,  $\mathcal{A}_1(l) := q$ , and  $\mathcal{A}_2(l) := r$ .
10:    if there exists  $k$  such that  $\mathcal{A}_1(l-k+1, \dots, l)$  and  $\mathcal{A}_2(l-k+1, \dots, l)$  form the
        same set then
11:      // Cycling is detected.
12:      cycle := true
13:    else
14:      cycle := false
15:    end if
16:  end if
17: end function

```

**Algorithm 3:** Cycle detection for Algorithm 2.

if and only if  $\hat{x}$  is a minimizer to  $\text{LP}(\hat{x}, \mathcal{P})$  for all  $\mathcal{P} \subseteq \mathcal{D}(\hat{x})$ . Otherwise, we can find a descent direction from some LP piece to leave the vertex  $\hat{x}$ .

The naive approach just described can be improved by the following observation. Applying Bland's rule to  $\text{LP}(\hat{x}, \mathcal{P})$  for some  $\mathcal{P} \subseteq \mathcal{D}(\hat{x})$ , assume that we obtain a set of multipliers  $\hat{y}_i$  that satisfies (8) and therefore  $\hat{x}$  is a solution to  $\text{LP}(\hat{x}, \mathcal{P})$ . If for some  $i \in \mathcal{P}$  we have  $\hat{y}_i \geq 0$  in addition to  $\hat{y}_{c(i)} \geq 0$ , then  $\hat{x}$  is also a solution to  $\text{LP}(\hat{x}, \mathcal{P} \setminus \{i\})$ . In general, given a set of multipliers satisfying (8) at  $\hat{x}$ , we let

$$\mathcal{R}_1 = \{i : \hat{y}_i \geq 0 \wedge i \in \mathcal{P}\}, \quad \mathcal{R}_2 = \{i : \hat{y}_{c(i)} \geq 0 \wedge i \in \mathcal{D}(\hat{x}) \setminus \mathcal{P}\}, \quad (20)$$

be the index sets corresponding to strongly stationary components. Then,  $\hat{x}$  is a solution to all the LP pieces

$$\text{LP}(\hat{x}, \mathcal{P} \cup \mathcal{S}_2 \setminus \mathcal{S}_1) \quad \text{for all} \quad \mathcal{S}_1 \subseteq \mathcal{R}_1, \quad \mathcal{S}_2 \subseteq \mathcal{R}_2. \quad (21)$$

This observation indicates that a set of multipliers can be used to detect the optimality of multiple LP pieces in Definition 2.3.

A simplistic implementation of our anticycling scheme is as follows. We set  $\mathcal{U}$  equal to  $2^{\mathcal{D}(\hat{x})}$  and remove  $\mathcal{P}$  from  $\mathcal{U}$  whenever  $\hat{x}$  is verified as a minimizer of  $\text{LP}(\hat{x}, \mathcal{P})$ . The process repeats until  $\mathcal{U} = \emptyset$ , in which case  $\hat{x}$  is a B-stationary point, or until we find a descent search direction to leave  $\hat{x}$ , in which case the cycle of pivoting breaks. The pseudo-code is given in Algorithm 4.

The condition (17) must still be satisfied during anticycling by Algorithm 4. This condition has implications for the two cases:

- (1) For each strict complementary condition, one constraint  $i$  is active and has been in the extended working set, that is,  $i \in \bar{\mathcal{W}} = \mathcal{W} \cup \mathcal{E}$ . This active constraint cannot move away from  $\bar{\mathcal{W}}$ , but it may move from  $\mathcal{E}$  to  $\mathcal{W}$ . The other complementarity constraint  $c(i)$  is inactive and therefore is not in the extended working set, that is,  $c(i) \notin \bar{\mathcal{W}}$ . Hence, we can ignore these strict complements when determining the leaving constraints.
- (2) For each nonstrict complementary condition, one complementarity constraint is treated as an equality and the other is treated as an inequality, with respect to

```

1: function [status,  $\mathcal{W}$ ,  $\mathcal{E}$ ,  $\hat{x}$ ] = ANTICYCLE( $\mathcal{W}$ ,  $\mathcal{E}$ ,  $\hat{x}$ )
2:   Set  $\mathcal{U} := 2^{\mathcal{D}(\hat{x})}$ , where  $\mathcal{D}(\hat{x}) = \{i : a_i^T \hat{x} = b_i \wedge a_{c(i)}^T \hat{x} = b_{c(i)}\}$ .
3:   repeat
4:     Select  $\mathcal{P} \in \mathcal{U}$ ; let  $\mathcal{P}^c := \{c(i) : i \in \mathcal{D}(\hat{x}) \setminus \mathcal{P}\}$ .
5:     // We consider LP( $\hat{x}, \mathcal{P}$ ) and apply Bland's least index rule.
6:      $\mathcal{E} := \mathcal{E} \cup \{i : i \in \mathcal{P} \cup \mathcal{P}^c \wedge i \notin \mathcal{W}\}$ 
7:     repeat
8:       Compute  $\hat{y} \equiv [\hat{y}_i]_{i \in \mathcal{W}} := A^{-1}g$ , where  $A := [a_j]_{j \in \mathcal{W}}$ .
9:       Compute  $q := \min\{i : i \in \mathcal{C}_0 \cup \mathcal{C}_1\}$ , where
10:         $\mathcal{C}_0 := \min\{i : i \in \mathcal{W}_0 \wedge \hat{y}_i < 0\}$ 
11:         $\mathcal{C}_1 := \min\{i : i \in \mathcal{W}_1 \setminus (\mathcal{P} \cup \mathcal{P}^c) \wedge \hat{y}_i < 0\}$ 
12:        // Here  $\mathcal{W}_0 = \mathcal{W} \cap \{1, \dots, m\}$  and  $\mathcal{W}_1 = \mathcal{W} \cap \{m+1, \dots, m+2p\}$ .
13:        if  $q \neq \text{NULL}$  then
14:           $\hat{y}_q < 0$  is the qualified multiplier with smallest index.
15:          Compute search direction  $s_q$ , the column of  $A^{-T}$  corresponding to  $\hat{y}_q$ .
16:          if  $\exists r \in \mathcal{E}$  such that  $a_r^T s_q \neq 0$  then
17:             $\mathcal{E} := \mathcal{E} \setminus \{r\}$ ;  $\mathcal{W} := \mathcal{W} \cup \{r\} \setminus \{q\}$ 
18:          else
19:            Ratio test:  $\hat{\alpha}_r := \min_{\substack{j \in \mathcal{W} \\ a_j^T s_q < 0}} \left\{ \frac{b_j - a_j^T \hat{x}}{a_j^T s_q}, \infty \right\}$ .
20:            If multiple choices of  $r$  exist, choose the smallest one.
21:            if  $\hat{\alpha}_r = \infty$  then
22:              // LP( $\hat{x}, \mathcal{P}$ ) is unbounded, so is the LPCC.
23:              return: status := unbounded.
24:            end if
25:             $\mathcal{W} := \mathcal{W} \cup \{r\} \setminus \{q\}$ 
26:            if  $\hat{\alpha}_r > 0$  then // cycle breaks.
27:               $\hat{x} = \hat{x} + \hat{\alpha}_r s_q$ 
28:              return: status := cycle.breaks.
29:            end if
30:          end if
31:        end if
32:      until  $q = \text{NULL}$  (i.e., no qualified leaving constraint).
33:      //  $\hat{x}$  is a solution to LP( $\hat{x}, \mathcal{P}$ ).
34:      Set  $\mathcal{R}_1 := \{i : \hat{y}_i \geq 0 \wedge i \in \mathcal{P}\}$ ,  $\mathcal{R}_2 := \{i : \hat{y}_{c(i)} \geq 0 \wedge i \in \mathcal{D}(\hat{x}) \setminus \mathcal{P}\}$ 
35:      Update  $\mathcal{U} := \mathcal{U} \setminus \{\mathcal{P} \cup \mathcal{S}_2 \setminus \mathcal{S}_1 : \mathcal{S}_1 \subseteq \mathcal{R}_1 \wedge \mathcal{S}_2 \subseteq \mathcal{R}_2\}$ 
36:      until  $\mathcal{U} = \emptyset$ .
37:      return: status := B_stationary.
38:    end function

```

**Algorithm 4:** Anticycling for Algorithm 2.

LP( $\hat{x}, \mathcal{P}$ ) in each inner loop. Those treated as equalities must be in  $\bar{\mathcal{W}}$ . In other words,

$$\mathcal{P} \cup \mathcal{P}^c \subseteq \bar{\mathcal{W}}, \quad \mathcal{P}^c = \{c(i) : i \in \mathcal{D}(\hat{x}) \setminus \mathcal{P}\}. \quad (22)$$

We can ensure (22) by requiring two conditions:

- At the beginning of each inner loop, we add the required constraints into  $\mathcal{E}$  to make  $\bar{\mathcal{W}} = \mathcal{W} \cup \mathcal{E}$  satisfy (22).
- At each pivoting step, we prevent the constraints in  $\mathcal{P} \cup \mathcal{P}^c$  from leaving  $\mathcal{W}$ .

In each inner loop of Algorithm 4, we consider LP( $\hat{x}, \mathcal{P}$ ) and apply Bland's least index

rule. Therefore, the inner loop must terminate in a finite number of iterations [10, Theorem 3.3]; [20, Theorem 8.3.1]. Note, however, that we do not need Bland's rule when moving a constraint from  $\mathcal{E}$  into  $\mathcal{W}$ , because in each inner loop, constraints in  $\mathcal{E}$  can leave but no constraint can enter  $\mathcal{E}$ , and therefore it cannot cause a cycling.

For the outer loop, the size of  $\mathcal{U}$  is reduced at each iteration. Therefore, the overall number of iterations is finite. We conclude that our anticycling scheme in Algorithm 4 must terminate with one of the following three cases:

- (1) A descent search direction is found, and there is no stopping constraint, that is,  $\hat{\alpha}_r = \infty$ . The current LP( $\hat{x}, \mathcal{P}$ ) of concern is unbounded, and so is the LPCC (1).
- (2) A descent direction is found, and we move to another vertex with a positive step length, in which case we go back to Algorithm 2 and continue with the next pivoting step.

**Remark:** Note that (18) may no longer hold because we augment  $\mathcal{E}$ . Nevertheless, (18) is not required by Algorithm 2 to work properly, but it potentially avoids unnecessary pivoting steps. Indeed, we can impose

$$\mathcal{E} := \{i : i \in \mathcal{E} \wedge c(i) \notin \mathcal{W} \wedge (i \leq m+p \vee c(i) \notin \mathcal{E})\}$$

when returning to Algorithm 2, if (18) is desired.

- (3) If  $\mathcal{U} = \emptyset$  at the termination of Algorithm 4, then  $\hat{x}$  is a minimizer to all LP( $\hat{x}, \mathcal{P}$ ) for  $\mathcal{P} \in \mathcal{D}(\hat{x})$ , and therefore B-stationary.

The discussion leads to the following proposition.

**PROPOSITION 5.1** *Algorithm 2 must terminate in a finite number of pivoting steps, either finding a descent direction to leave the current vertex  $\hat{x}$  or verifying that  $\hat{x}$  is B-stationary.*

We are free to choose any  $\mathcal{P} \in \mathcal{U}$  in Algorithm 4. If we carefully select  $\mathcal{P} \in \mathcal{U}$ , the number of pivoting steps may be reduced, as the following illustrative example shows. We apply Algorithm 2 to solve the LPCC (9) which has only one vertex  $\hat{x} = (0, 0, 0)$ . We note that in this example the pivoting methods finds the global minimum, but does not provide a certificate of global optimality, which would require a tree-search.

**The first pivoting step:** Suppose we have the initial working set  $\mathcal{W} = \{1, 2, 3\}$  with zero extension  $\mathcal{E} = \emptyset$ . The corresponding multipliers  $\hat{y} \equiv [\hat{y}_j]_{j \in \mathcal{W}} = A^{-1}g$  are  $(\hat{y}_1, \hat{y}_2, \hat{y}_3) = (\frac{3}{4}, \frac{1}{4}, -2)$ , where  $\hat{y}_3 = -2$  is the only negative multiplier, associated with constraint 3, which is complementary. The other complement, indexed by 4, is active but not in  $\bar{\mathcal{W}} = \mathcal{W} \cup \mathcal{E}$ . Therefore the leaving constraint  $q$  is 3, and we add constraint 4 into  $\mathcal{E}$ , resulting in  $\mathcal{E} = \{4\}$ . The search direction  $s_q$  is  $(1, 1, 4)$ . We move constraint 4 from  $\mathcal{E}$  to  $\mathcal{W}$  immediately, since  $a_4 s_q = 1 \neq 0$ , where  $a_4 = (0, 1, 0)$  is the gradient of constraint 4. In Algorithm 3 for cycle detection, the updated arrays are  $\mathcal{A}_1 = (3)$  and  $\mathcal{A}_2 = (4)$ .

**The second pivoting step:** The working set  $\mathcal{W}$  is now  $\{1, 2, 4\}$ , associated with multipliers  $(\hat{y}_1, \hat{y}_2, \hat{y}_4) = (\frac{1}{4}, \frac{3}{4}, -2)$ . With a similar discussion, the leaving and entering constraints are those indexed by 4 and 3, respectively. At this point we detect a cycle by Algorithm 3, where the updated arrays are  $\mathcal{A}_1 = (3, 4)$  and  $\mathcal{A}_2 = (4, 3)$ , forming the same set  $\{3, 4\}$ . Hence, we do not pivot but instead move on to Algorithm 4 for anticycling.

**The anticycling phase:** When entering Algorithm 4, we have the working set  $\mathcal{W} = \{1, 2, 4\}$ . The only complementarity condition is degenerate, so  $\mathcal{D}(\hat{x}) = \{3\}$ . We initialize  $\mathcal{U} := \{\emptyset, \{3\}\}$  as the power set of  $\mathcal{D}(\hat{x})$ . Now we select  $\mathcal{P}$  to be  $\emptyset \in \mathcal{U}$ ; then  $\mathcal{P}^c = \{4\}$ , and  $\mathcal{E}$

remains empty. The only negative multiplier associated with  $\mathcal{W} = \{1, 2, 4\}$  is  $\hat{y}_4 = -2$ . Since  $4 \in \mathcal{P} \cup \mathcal{P}^c$ , constraint 4 is treated as equality in  $\text{LP}(\hat{x}, \emptyset)$  and cannot leave  $\mathcal{W}$ . Therefore,  $\hat{x}$  is a minimizer of  $\text{LP}(\hat{x}, \emptyset)$ . The updated  $\mathcal{U}$  contains only one element  $\{3\}$  after removing  $\emptyset$ .

We continue with the next outer iteration of Algorithm 4. The next  $\mathcal{P}$  is  $\{3\}$ , whose corresponding  $\mathcal{P}^c$  is  $\emptyset$ . Since  $3 \in \mathcal{P} \cup \mathcal{P}^c$  is not in the working set  $\mathcal{W} = \{1, 2, 4\}$ , we add it into  $\mathcal{E}$  and obtain  $\mathcal{E} = \{3\}$ . The only negative multiplier is  $\hat{y}_4 = -2$ . Since now the LP piece of concern is  $\text{LP}(\hat{x}, \{3\})$ , constraint 4 is treated as an inequality and can leave the working set  $\mathcal{W}$ . Then the entering constraint is  $3 \in \mathcal{E}$ . After pivoting, we have  $\mathcal{W} = \{1, 2, 3\}$  and  $\mathcal{E} = \emptyset$ . The only negative multiplier is  $\hat{y}_3 = -2$ . However, constraint 3 is treated as an equality in  $\text{LP}(\hat{x}, \{3\})$ , so it cannot leave the working set  $\mathcal{W}$ . Hence  $\hat{x} = (0, 0, 0)$  is also a minimizer of  $\text{LP}(\hat{x}, \{3\})$ . We conclude that  $\hat{x} = (0, 0, 0)$  is a B-stationary point.

**Discussion:** In the above illustration, we select  $\mathcal{P}$  to be  $\emptyset$  for the first inner loop of Algorithm 4. Alternatively, if we choose  $\mathcal{P}$  to be  $\{3\}$ , then the set  $\mathcal{E}$  will be augmented to be  $\{3\}$ . As a result, the first LP piece requires one more pivoting step to move constraint 3 from  $\mathcal{E}$  to  $\mathcal{W}$ . Careful selection of  $\mathcal{P} \subseteq \mathcal{U}$  may save more pivots for larger programs.

Now we discuss how to select  $\mathcal{P} \subseteq \mathcal{U}$  in Algorithm 4. Adding constraints into  $\mathcal{E}$  will potentially increase the number of pivoting steps to move constraints from  $\mathcal{E}$  to  $\mathcal{W}$ . Therefore, the key point is to constrain the augmentation of  $\mathcal{E}$ . Recall that the purpose of augmenting  $\mathcal{E}$  is to satisfy (22). As discussed in Section 4, our  $\mathcal{W}$  in Algorithm 2 satisfies (17), so there exists  $\mathcal{P}$  satisfying (22). Hence augmentation of  $\mathcal{E}$  is not required for the first inner loop. A greedy method is to select the  $\mathcal{P} \in \mathcal{U}$  that is closest to the previous one, denoted by  $\hat{\mathcal{P}}$ , for each consequent inner loop. In other words,

$$\mathcal{P} := \operatorname{argmax}\{|\mathcal{P} \cap \hat{\mathcal{P}}| : \mathcal{P} \in \mathcal{U}\}.$$

## 6. Obtaining an Initial LPCC Feasible Vertex

Our pivoting algorithm to solve an LPCC (1) requires a feasible starting vertex. Similar to the Phase I process of the simplex method for linear programming, we propose a two-phase process for LPCC (1).

Our Phase I is identical to the Phase I of LP, where we find a linear feasible vertex satisfying all the inequalities (3). If such a vertex cannot be found, then the inequalities (3) cannot be satisfied, and the LPCC (1) is *globally infeasible*, because the LP relaxation of the LPCC (obtained by relaxing the complementarity conditions) is infeasible. If such a vertex is found, we continue with Phase II to resolve complementary violations in order to satisfy (4). If a feasible point  $\hat{x}$  is found, we move on to the optimality phase, where we use  $\hat{x}$  as the starting vertex to solve LPCC (1) by our Algorithm 2.

In Phase II we have a linearly feasible vertex  $\hat{x}$  from Phase I, but some of the complementarity conditions may not be satisfied. We note that Phase I ensures that  $a_i^T \hat{x} - b_i \geq 0$  and  $a_{c(i)}^T \hat{x} - b_{c(i)} \geq 0$ , which implies  $(a_i^T \hat{x} - b_i)(a_{c(i)}^T \hat{x} - b_{c(i)}) \geq 0$ . Therefore, we partition the complementarity conditions into two sets of  $\mathcal{I}$  and  $\mathcal{J}$ , of those satisfied and of those not yet satisfied, respectively:

$$\begin{aligned} \mathcal{I} &= \{i : (a_i^T \hat{x} - b_i)(a_{c(i)}^T \hat{x} - b_{c(i)}) = 0, i = m+1, \dots, m+p\}, \\ \mathcal{J} &= \{i : (a_i^T \hat{x} - b_i)(a_{c(i)}^T \hat{x} - b_{c(i)}) > 0, i = m+1, \dots, m+p\}. \end{aligned} \quad (23)$$

We resolve the complementary violation one at a time, and update (23) at each iteration. The pseudo-code is given in Algorithm 5.

- 1: // Given a linearly feasible vertex  $\hat{x}$  of LPCC (1) from Phase I.
- 2: Determine  $\mathcal{I}$  and  $\mathcal{J}$  by (23).
- 3: Set  $\mathcal{K} := \emptyset$
- 4: **repeat**
- 5:     Select  $j \in \mathcal{J} \setminus \mathcal{K}$ . Starting from  $\hat{x}$ , solve the following reduced LPCC:

$$\begin{cases} \underset{x}{\text{minimize}} & a_j^T x \\ \text{subject to} & a_i^T x \geq b_i, \quad i = 1, \dots, m+2p, \\ & 0 \leq (a_i^T x - b_i) \perp (a_{c(i)}^T x - b_{c(i)}) \geq 0, \quad i \in \mathcal{I}. \end{cases} \quad (24)$$

- 6:     **if** the minimizer  $x^*$  of (24) satisfies  $a_j^T x^* = b_j$  **then**
- 7:          $\mathcal{K} := \emptyset$ ;  $\hat{x} := x^*$ ; update  $\mathcal{I}$  and  $\mathcal{J}$  by (23).
- 8:     **else**
- 9:         Starting from  $\hat{x}$ , solve the LPCC:

$$\begin{cases} \underset{x}{\text{minimize}} & a_{c(j)}^T x \\ \text{subject to} & a_i^T x \geq b_i, \quad i = 1, \dots, m+2p, \\ & 0 \leq (a_i^T x - b_i) \perp (a_{c(i)}^T x - b_{c(i)}) \geq 0, \quad i \in \mathcal{I}. \end{cases} \quad (25)$$

- 10:     **if** the minimizer  $x^*$  of (25) satisfies  $a_{c(j)}^T x^* = b_{c(j)}$  **then**
- 11:          $\mathcal{K} := \emptyset$ ;  $\hat{x} := x^*$ ; update  $\mathcal{I}$  and  $\mathcal{J}$  by (23).
- 12:     **else**
- 13:          $\mathcal{K} := \mathcal{K} \cup \{j\}$
- 14:     **end if**
- 15:     **end if**
- 16: **until**  $\mathcal{J} = \emptyset$  or  $\mathcal{K} = \mathcal{J}$ .
- 17: // If  $\mathcal{J} = \emptyset$ , then the last  $\hat{x}$  is feasible.

**Algorithm 5:** Phase II to find a complementary feasible vertex of LPCC (1).

We note that we can solve LPCCs (24) and (25) using our LPCC pivoting scheme, because we have a feasible starting vertex. We also note the following:

- (1) When updating  $\hat{x} := x^*$ , we also update the extended working set  $\bar{\mathcal{W}} = \mathcal{W} \cup \mathcal{E}$ , inherited from (24) or (25).
- (2) Since we have at least one more satisfied complementarity condition,  $\mathcal{I}$  is augmented. After augmentation of  $\mathcal{I}$ , the condition (17) may no longer hold. Let

$$\mathcal{F} = \{i : i \in \mathcal{I} \wedge i \notin \bar{\mathcal{W}} \wedge c(i) \notin \bar{\mathcal{W}}\}$$

be the set of complementarity conditions without complements in the extended working set  $\bar{\mathcal{W}} = \mathcal{W} \cup \mathcal{E}$ . In order to satisfy (17), we add some complements to  $\mathcal{E}$ , those in

$$\Delta\mathcal{E} = \{i : i \in \mathcal{F} \wedge a_i^T \hat{x} = b_i\} \cup \{c(i) : i \in \mathcal{F} \wedge a_{c(i)}^T \hat{x} = b_{c(i)} \wedge a_i^T \hat{x} \neq b_i\}.$$

After augmenting  $\mathcal{E} := \mathcal{E} \cup \Delta\mathcal{E}$ , the resulting extended working set  $\bar{\mathcal{W}} = \mathcal{W} \cup \mathcal{E}$  satisfies (17). Finally, the augmentation of  $\mathcal{E}$  does not violate (18).

In Algorithm 5, either the size of  $\mathcal{J}$  is reduced after every update, or the current complementarity condition is added to  $\mathcal{K}$ . Therefore, Algorithm 5 must terminate in a finite number of iterations with one of two possible outcomes:

- (1)  $\mathcal{J} = \emptyset$ : We obtain a complementary feasible vertex  $\hat{x}$ . Then we continue with

the optimality phase, solving the LPCC (1) by our pivoting algorithm with the starting vertex  $\hat{x}$ .

- (2)  $\mathcal{K} = \mathcal{J} \neq \emptyset$ : We are not able to resolve the complementarity violations still in  $\mathcal{J}$ , in which case we call the LPCC (1) *locally infeasible*.

For every  $j \in \mathcal{J}$ , we can either solve (25) first or solve (24) first. In our implementation, we project the current vertex  $\hat{x}$  to the two subspaces formed by  $a_j^T x = b_j$  and  $a_{c(j)}^T x = b_{c(j)}$ , respectively. We solve (24) or (25) first depending on which projected point results in smaller objective function value.

**Note:** We can stop Algorithm 5 early if we reach line 13 where  $\mathcal{K} \neq \emptyset$ , which corresponds to a local minimum of LPCC constraint violation. On the other hand,  $\mathcal{K} \neq \emptyset$  does not imply that the problem is infeasible, because the feasible set is nonconvex. Continuing to solve LPCCs as presented in Algorithm 5 may find a feasible point of the LPCC, even after  $\mathcal{K} \neq \emptyset$  was reached. In our experiments on 168 LPCCs in Section 7, there are three problems where the infeasibility is resolved because of continuing after  $\mathcal{K} \neq \emptyset$  in line 13. These problems are `ex9.1.6-lpcc`, `tollmpec1-siouxfls-lpcc`, and `tollmpec-siouxfls-lpcc`.

We illustrate our approach with the LPCC (14). Starting with the feasible vertex  $\hat{x} = (0, 0, 0, 1, 0)$  associated with the working set  $\mathcal{W} = \{1, 2, 3, 5, 7\}$  determined in Phase I, we now resolve the complementarity violation by Algorithm 5. The only complementarity condition not yet satisfied is  $0 \leq x_4 \perp x_1 - x_3 + 2 \geq 0$ , for which the LPCC program (24) reads as

$$\left\{ \begin{array}{ll} \text{minimize } x_4 & \\ x_1, x_2, x_3, x_4, x_5 & \\ \text{subject to } x_1, x_2 \geq 0, & \text{indexed by 1,2;} \\ x_1 + 2x_4 \geq 2, & \text{indexed by 3;} \\ x_3 - x_4 - x_5 \geq -2, & \text{indexed by 4;} \\ 0 \leq x_3 \perp x_1 - x_2 + x_3 + 1 \geq 0, & \text{indexed by 5 and 8;} \\ 0 \leq x_4, \quad x_1 - x_3 + 2 \geq 0, & \text{indexed by 6 and 9;} \\ 0 \leq x_5 \perp x_3 - x_4 + 1 \geq 0, & \text{indexed by 7 and 10.} \end{array} \right. \quad (26)$$

The multipliers  $\hat{y} = [\hat{y}_j]_{j \in \mathcal{W}} = A^{-1}g$ , with  $g = (0, 0, 0, 1, 0)$  the objective normal of (26), are  $(y_1, y_2, y_3, y_5, y_7) = (-\frac{1}{2}, 0, \frac{1}{2}, 0, 0)$ . Therefore, the standard constraint  $x_1 \geq 0$  associated with the only negative multiplier  $\hat{y}_1 = -\frac{1}{2}$  is the leaving constraint. The entering constraint determined by the ratio test (13) is  $x_4 \geq 0$ , indexed by 6. So the updated working set is  $\mathcal{W} = \{2, 3, 5, 6, 7\}$ , and the vertex  $\hat{x} = (2, 0, 0, 0, 0)$  is feasible.

**Global Optimality of the Phase I/Phase II Approach.** We note that the proposed practical Phase I/II method may fail to find an initial feasible vertex, and get trapped in a local minimum of the constraint violation. For our purposes, this outcome is satisfactory since we are interested only in computing local solutions. For certain classes of LPCCs, however, we can easily find an initial feasible vertex or prove that none exists. One such class of problems are bilevel LPs, where the lower level problem is feasible for every choice of the upper level variables, and the upper level constraints involve only the upper level variables.

We have a MATLAB implementation of our pivoting algorithm, which can handle more general forms of linear constraints, including equality constraints, range constraints, and mixed complementarity conditions. We compare our solver with the filter MPEC solver via the NLP reformulation [16] that solves MPEC by introducing slack variables  $s$  and replacing complementarity conditions  $y \perp s$  by  $y^T s \leq 0$ . Filter MPEC uses an SQP method to solve the resulting NLP. We compare the two solvers on a set of LPCCs obtained by linearizing the MPECs from MacMPEC [34], a collection of 168 MPEC test problems written in AMPL [18]. AMPL allows more general constraints, including range constraints and equations. AMPL also allows mixed complementarity constraints. See Ferris et al. [14] for information of expressing complementarity conditions in AMPL.

Each LPCC can be characterized by the following numbers: the number of variables  $n$ , the number of constraints  $m$ , the number of equalities  $m_e$ , and the number of complementarity conditions  $p$ . Appendix A shows the characteristics of the 168 programs in MacMPEC-LPCC test set, where we have sorted the programs by  $n$ . We have removed 12 LPCCs from the test set, which are LPs after AMPL’s presolve eliminated all complementarity constraints. These problems are `bard3-lpcc`, `bard3m-lpcc`, and `gnash1j-lpcc` for  $j = 0, 1, \dots, 9$ .

Five outcomes are possible for our pivoting algorithm: globally infeasible, locally infeasible, B-stationary, strongly stationary, and unbounded objective. In practice, problems are expected during anticycling if we have a large set of nonstrict complementarity conditions  $\mathcal{D}(\hat{x})$ , defined in (5). Recall that Algorithm 4 for anticycling uses a set  $\mathcal{U}$  to track the determination of optimality of LP pieces  $\text{LP}(\hat{x}, \mathcal{P})$  for  $\mathcal{P} \subseteq \mathcal{D}(\hat{x})$ . The set  $\mathcal{U}$  is initialized as the power set of  $\mathcal{D}(\hat{x})$ , which is of size  $2^{|\mathcal{D}(\hat{x})|}$ . It means  $|\mathcal{D}(\hat{x})|$ , the size of set  $\mathcal{D}(\hat{x})$ , cannot be large in practice. We call  $|\mathcal{D}(\hat{x})|$  the *degree of nonstrictness* and allow  $|\mathcal{D}(\hat{x})| \leq 16$  in our experiments.

Table 1. Result summary of the 168 programs in MacMPEC-LPCC test set.

Pivoting Algorithm		Filter MPEC	
Outcome	# Programs	Outcome	# Programs
globally infeasible	21	linear infeasible	21
locally infeasible	2	locally infeasible	2
unbounded objective	30	unbounded objective	15
strongly stationary	111	optimal solution found	112
B-stationary	2	trust region too small	5
max deg. nonstrict. reached	2	max # iter. reached	13

We implemented a one-at-a-time Phase I method to get a linear feasible point. The results are summarized in Table 1. We can see from Table 1 that the results are consistent except for the following

- (1) In two cases the degree of nonstrictness exceeded 16, but filter MPEC found solutions. These problems are `monteiroB-lpcc` and `monteiro-lpcc`.
- (2) In 5 cases filter MPEC stopped because the trust region was smaller than its tolerance, which we had set as  $10^{-6}$ : `flp4-1-lpcc`, `flp4-3-lpcc`, `incid-set2-32-lpcc`, `incid-set2c-32-lpcc`, and `pack-rig1p-32-lpcc`.
- (3) In 13 cases filter MPEC was not able to solve within the maximum number of iterations, which we had set as 1000: `dempe-lpcc`, `design-cent-31-lpcc`, `design-cent-3-lpcc`, `liswet1-050-lpcc`, `liswet1-100-lpcc`, `liswet1-200-lpcc`, `outrata34-lpcc`, `qpecgen-100-1-lpcc`, `qpecgen-100-2-lpcc`, `qpecgen-100-3-lpcc`, `qpecgen-100-4-lpcc`, `qpecgen-200-1-lpcc`, and `qpecgen-200-2-lpcc`.

Figure 1 plots the numbers of pivoting steps in Phase I, Phase II, and Phase III, for the 168 LPCCs. We sorted the LPCCs by the number of variables  $n$ . As expected, larger programs tend to take more pivoting steps to solve. On the other hand, no phase dominated the computational cost in all cases. Detailed results are given in Tables A1–A4.

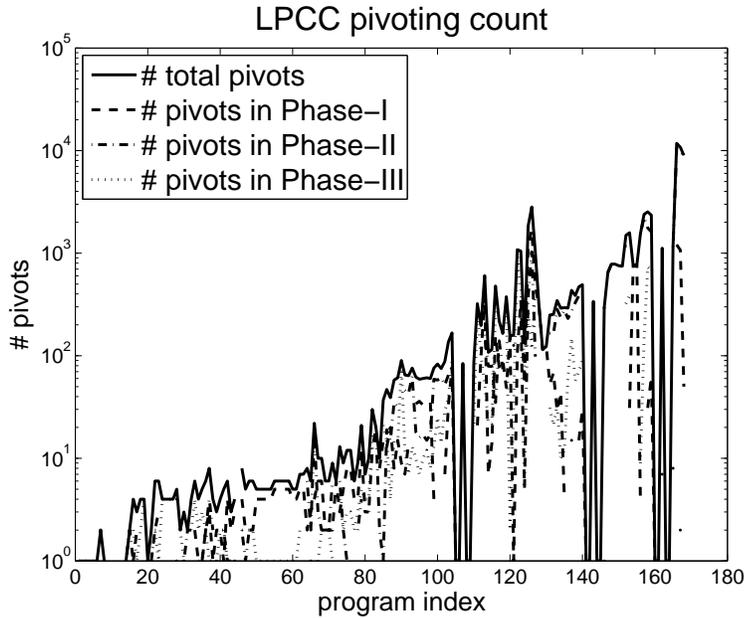


Figure 1. Pivoting counts of our pivoting algorithm using the MacMPEC-LPCC test set.

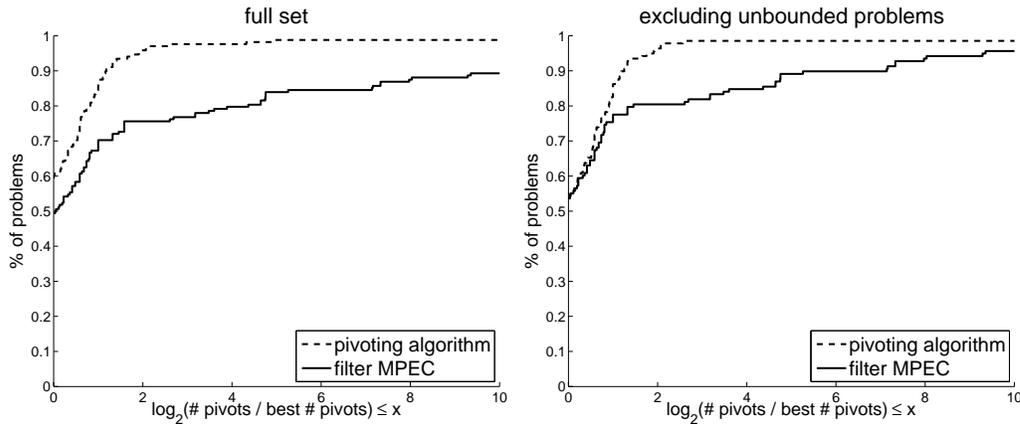


Figure 2. Performance profiles ( $\log_2$  scale) using the MacMPEC-LPCC test set. Left: all 168 problems; right: excluding 30 unbounded problems.

Figure 2 shows the performance profiles [11, 12] of our pivoting algorithm and filter MPEC. In the left plot we use all 168 test problems in the MacMPEC-LPCC test set. The plot indicates that our pivoting algorithm outperforms filter MPEC significantly. We note that filter MPEC sometimes takes orders of magnitude more pivoting steps than does our pivoting pivoting algorithm to verify the unbounded objectives. The reason may be that filter MPEC solves nonlinear MPECs and cannot take advantage of the possibility of unbounded linear rays. Therefore, in the right plot we remove the 30 unbounded problems. The results indicate that our pivoting algorithm still performs better.

## 8. Conclusion

We give a pivoting algorithm to solve linear programs with linear complementarity constraints. Our algorithm is based on the active set method for linear programming. It works under degeneracy and includes an anticycling scheme that can determine B-stationarity and avoid infinite loops. We also use an optimization-based technique that consists of two phases to find an initial feasible vertex. Phase I is to find a linear feasible vertex, whereas Phase II is to resolve complementary violations. The experimental results indicate that our method is an appealing alternative to existing techniques.

## Acknowledgments

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357. This work was also supported by NSF grant 0631622.

## Appendix A. MacMPEC-LPCC Program Characteristics

Tables A1–A4 list the following characteristic numbers of the 168 LPCCs in the MacMPEC-LPCC test set: the number of variables  $n$ , the number of constraints  $m$ , the number of equalities  $m_e$ , and the number of complementarity conditions  $p$ . The numbers of pivots of our pivoting algorithm and filter MPEC for each LPCC are also listed. Three error codes are used in these tables: “(d)” for the maximum degree of nonstrictness reached, “(i)” for the maximum number of iterations reached, and “(t)” for that trust region is too small.

## References

- [1] Anitescu, M. (2005). On using the elastic mode in nonlinear programming approaches to mathematical programs with complementarity constraints. *SIAM J. Optim.*, 15:1203–1236.
- [2] Anitescu, M., Tseng, P., and Wright, S. J. (2007). Elastic-mode algorithms for mathematical programs with equilibrium constraints. *Math. Program.*, 110:337–371.
- [3] Audet, C., Hansen, P., Jaumard, B., and Savard, G. (1997). Links between linear bilevel and mixed 0-1 programming problems. *J. Optim. Theory Appl.*, 93:273–300.
- [4] Audet, C., Savard, S., and Zghal, W. (2007). New branch-and-cut algorithm for bilevel linear programming. *J. Optim. Theory Appl.*, 134:353–370.
- [5] Bard, J. F. and Moore, J. T. (1990). A branch and bound algorithm for the bilevel programming problem. *SIAM J. Sci. Comput.*, 11(2):281–292.
- [6] Bartels, R. H. and Golub, G. H. (1969a). Algorithm 350: Simplex method procedure employing  $LU$  decomposition. *Commun. of ACM*, 12:275–281.
- [7] Bartels, R. H. and Golub, G. H. (1969b). The simplex method of linear programming using  $LU$  decomposition. *Commun. of ACM*, 12:266–268.
- [8] Benson, H., Sen, A., Shanno, D. F., and Vanderbei, R. V. D. (2006). Interior-point algorithms, penalty methods and equilibrium problems. *Comput. Optim. Appl.*, 34(2):155–182.
- [9] Campêlo, M. and Scheimberg, S. (2000). A note on a modified simplex approach for solving bilevel linear programming problems. *European J. Oper. Res.*, 126:454–458.
- [10] Chvátal, V. (1983). *Linear Programming*. W. H. Freeman & Company.
- [11] Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Math. Program. Ser. A*, 91:201–213.
- [12] Dolan, E. D., Moré, J. J., and Munson, T. (2006). Optimality measures for performance profiles. *SIAM J. Optim.*, 16(3):891–909.
- [13] Facchinei, F., Jiang, H., and Qi, L. (1999). A smoothing method for mathematical programs with equilibrium constraints. *Math. Program.*, 85:107–134.
- [14] Ferris, M. C., Fourer, R., and Gay, D. M. (1999). Expressing complementarity problems in an algebraic modeling language and communicating them to solvers. *SIAM J. Optim.*, 9:991–1009.
- [15] Fletcher, R. and Leyffer, S. (2004). Solving mathematical program with complementarity constraints as nonlinear programs. *Optim. Methods Soft.*, 19(1):15–40.
- [16] Fletcher, R., Leyffer, S., Ralph, D., and Scholtes, S. (2006). Local convergence of SQP methods for mathematical programs with equilibrium constraints. *SIAM J. Optim.*, 17(1):259–286.

Table A1. Results of the 168 programs in MacMPEC-LPCC test set, Part I.

Program	$n$	$m$	$m_e$	$p$	Pivoting Algor.		Filter MPEC	
					# Pivots	Min.	# Pivots	Min.
bard1-lpcc	5	9	1	3	6	-16	8	-16
bard1m-lpcc	6	10	1	3	5	19	8	-16
bard2-lpcc	12	21	5	3	7	-7743.0	8	-7743.0
bard2m-lpcc	12	21	5	3	7	-7743.0	9	-7743.0
bar-truss-3-lpcc	35	45	28	6	10	infeasible	15	infeasible
bilevel1-lpcc	10	17	2	6	8	5	8	0
bilevel1m-lpcc	8	13	2	4	6	5	12	5
bilevel2-lpcc	16	29	4	8	12	-7733.3	20	-7743.0
bilevel2m-lpcc	16	29	4	8	12	-7733.3	20	-7743.0
bilevel3-lpcc	10	14	6	2	5	-13.5	3	-13.5
bilin-lpcc	8	15	0	6	8	-13	165	-5.6
bem-milanc30-s-lpcc	3436	4901	1968	1464	8985	170.55	1517	170.55
dempe-lpcc	3	3	1	1	2	unbounded	3	(i)
design-cent-1-lpcc	12	15	6	3	8	-1.6794	97	-1.6794
design-cent-21-lpcc	13	19	6	3	10	-10.673	9	-10.673
design-cent-2-lpcc	13	19	6	3	10	-10.673	10	-10.673
design-cent-31-lpcc	15	15	6	3	13	unbounded	65	(i)
design-cent-3-lpcc	15	15	6	3	9	unbounded	63	(i)
design-cent-4-lpcc	22	33	10	8	7	-4	7	-4
desilva-lpcc	6	8	2	2	2	-8	0	-8
df1-lpcc	2	2	0	1	1	-3	0	-3
ex9.1.10-lpcc	11	17	5	3	5	-3.25	4	-3.25
ex9.1.1-lpcc	13	18	7	5	6	-13	5	-13
ex9.1.2-lpcc	8	13	5	2	4	-3	2	-6.25
ex9.1.3-lpcc	23	35	15	6	10	-23	8	-29.2
ex9.1.4-lpcc	8	13	5	2	3	-37	3	-37
ex9.1.5-lpcc	13	20	7	5	6	-1	6	-1
ex9.1.6-lpcc	14	21	7	6	9	-15	8	-21
ex9.1.7-lpcc	17	26	9	6	9	-6	7	-23
ex9.1.8-lpcc	11	17	5	3	5	-3.25	4	-3.25
ex9.1.9-lpcc	12	18	6	5	6	9.2	7	3.1111
ex9.2.1-lpcc	10	15	5	4	6	-16	7	-16
ex9.2.2-lpcc	9	14	4	3	3	-100	6	-100
ex9.2.3-lpcc	14	23	8	4	6	5	6	5
ex9.2.4-lpcc	8	12	5	2	4	unbounded	4	unbounded
ex9.2.5-lpcc	8	11	4	3	5	-47	7	-47
ex9.2.6-lpcc	16	22	6	6	6	unbounded	4	unbounded
ex9.2.7-lpcc	10	15	5	4	6	-16	7	-16
ex9.2.8-lpcc	6	9	3	2	3	1.5	4	1.5
ex9.2.9-lpcc	9	14	5	3	4	2	2	2
flp2-lpcc	4	5	0	2	6	-375	8	-375
flp4-1-lpcc	80	90	0	30	90	unbounded	2829	(t)
flp4-2-lpcc	110	170	0	60	322	-25.636	211	-25.636
flp4-3-lpcc	140	240	0	70	376	unbounded	970	(t)
flp4-4-lpcc	200	350	0	100	1081	-51.150	548	-51.150
gauvin-lpcc	3	4	0	2	4	-256.25	5	-256.25
gnash10m-lpcc	10	15	5	4	5	-5325.6	9	-5325.6
gnash11m-lpcc	10	15	5	4	5	-3428.5	8	-3428.5202

Table A2. Results of the 168 programs in MacMPEC-LPCC test set, Part II.

Program	$n$	$m$	$m_e$	$p$	Pivoting Algor.		Filter MPEC	
					# Pivots	Min.	# Pivots	Min.
gnash12m-lpcc	10	15	5	4	5	-1784.9	2	-1784.9
gnash13m-lpcc	10	15	5	4	5	-1147.7	2	-1147.7
gnash14m-lpcc	10	15	5	4	5	-847.42	2	-847.42
gnash15m-lpcc	10	15	5	4	6	-5325.6	9	-5325.6
gnash16m-lpcc	10	15	5	4	6	-3428.5	9	-3428.5
gnash17m-lpcc	10	15	5	4	6	-1784.9	4	-1784.9
gnash18m-lpcc	10	15	5	4	6	-1147.7	4	-1147.7
gnash19m-lpcc	10	15	5	4	6	-847.42	4	-847.42
hakonsen-lpcc	9	17	3	4	0	infeasible	0	infeasible
hs044-i-lpcc	20	30	4	10	21	-1.0693	23	-8.6036
incid-set1-16-lpcc	371	637	225	111	292	0	205	9.28E-17
incid-set1-32-lpcc	1517	2559	961	489	1492	0.000002	935	0.000002
incid-set1-8-lpcc	100	170	49	32	75	0	62	6.59E-17
incid-set1c-16-lpcc	371	652	225	111	435	0.000353	237	0.000353
incid-set1c-32-lpcc	1517	2590	961	489	1583	0.000249	824	0.000249
incid-set1c-8-lpcc	100	177	49	32	88	0	78	6.59E-17
incid-set2-16-lpcc	450	681	225	190	471	0.299364	18038	0.299364
incid-set2-32-lpcc	1857	2767	961	829	2532	0.299167	101792	(t)
incid-set2-8-lpcc	112	177	49	44	108	0.302232	975	0.302232
incid-set2c-16-lpcc	450	696	225	190	493	0.300847	13251	0.300714
incid-set2c-32-lpcc	1857	2798	961	829	2349	0.301222	28910	(t)
incid-set2c-8-lpcc	112	184	49	44	115	0.302271	742	0.302271
jr1-lpcc	2	1	0	1	1	unbounded	0	unbounded
jr2-lpcc	2	1	0	1	1	unbounded	0	unbounded
kth1-lpcc	2	1	0	1	1	0	0	0
kth2-lpcc	2	1	0	1	1	unbounded	1	unbounded
kth3-lpcc	2	1	0	1	1	0	1	0
liswet1-inv-050-lpcc	152	203	52	50	155	unbounded	68	(i)
liswet1-inv-100-lpcc	302	403	102	100	309	-50.864	140	(i)
liswet1-inv-200-lpcc	602	803	202	200	647	-100.53	360	(i)
monteiroB-lpcc	131	226	57	57	218	(d)	298	-3615.6
monteiro-lpcc	131	226	57	57	165	(d)	136	-2606.9
nash1a-lpcc	6	8	2	2	2	0	5	0
nash1b-lpcc	6	8	2	2	4	-95	10	-95
nash1c-lpcc	6	8	2	2	6	-240	12	-240
nash1d-lpcc	6	8	2	2	4	-53.3333	11	-100
nash1e-lpcc	6	8	2	2	5	-140	7	-140
outrata31-lpcc	5	8	0	4	4	-52.467	7	-52.467
outrata32-lpcc	5	8	0	4	4	-56.973	7	-56.973
outrata33-lpcc	5	8	0	4	4	-52.467	7	-52.467
outrata34-lpcc	5	8	0	4	4	-56.473	795	(i)
pack-comp1-16-lpcc	467	753	225	225	1	infeasible	143	infeasible
pack-comp1-32-lpcc	1955	3101	961	961	1	infeasible	624	infeasible
pack-comp1-8-lpcc	107	179	49	49	1	infeasible	27	infeasible
pack-comp1c-16-lpcc	467	768	225	225	1	infeasible	141	infeasible
pack-comp1c-32-lpcc	1955	3132	961	961	1	infeasible	655	infeasible
pack-comp1c-8-lpcc	107	186	49	49	1	infeasible	27	infeasible
pack-comp1p-16-lpcc	467	708	225	225	339	390.47	347	390.47

Table A.3. Results of the 168 programs in MacMPEC-LPCC test set, Part III.

Program	$n$	$m$	$m_e$	$p$	Pivoting Algor.		Filter MPEC	
					# Pivots	Min.	# Pivots	Min.
pack-comp1p-32-lpcc	1955	2948	961	961	1115	201.78	1049	201.78
pack-comp1p-8-lpcc	107	164	49	49	84	884.69	83	884.69
pack-comp2-16-lpcc	467	753	225	225	1	infeasible	161	infeasible
pack-comp2-32-lpcc	1955	3101	961	961	1	infeasible	262	infeasible
pack-comp2-8-lpcc	107	179	49	49	1	infeasible	25	infeasible
pack-comp2c-16-lpcc	467	768	225	225	1	infeasible	161	infeasible
pack-comp2c-32-lpcc	1955	3132	961	961	1	infeasible	251	infeasible
pack-comp2c-8-lpcc	107	186	49	49	1	infeasible	25	infeasible
pack-comp2p-16-lpcc	467	708	225	225	294	884.99	293	884.99
pack-comp2p-32-lpcc	1955	2948	961	961	1159	763.31	1003	763.31
pack-comp2p-8-lpcc	107	164	49	49	86	1953.7	93	1953.7
pack-rig1-16-lpcc	333	511	204	82	250	0.6	185	0.6
pack-rig1-32-lpcc	1433	2171	856	505	781	infeasible	186	infeasible
pack-rig1-8-lpcc	70	109	46	9	47	0.6	32	0.6
pack-rig1c-16-lpcc	333	526	204	82	253	0.6	174	0.6
pack-rig1c-32-lpcc	1433	2202	856	505	781	infeasible	186	infeasible
pack-rig1c-8-lpcc	70	116	46	9	39	0.6	26	0.6
pack-rig1p-16-lpcc	389	580	225	147	386	0.773051	240	0.773051
pack-rig1p-32-lpcc	1711	2571	961	717	2387	0.82143	7443	(t)
pack-rig1p-8-lpcc	92	138	49	34	83	0.733602	72	0.733602
pack-rig2-16-lpcc	326	510	204	93	115	infeasible	65	infeasible
pack-rig2-32-lpcc	1580	2694	856	661	756	infeasible	201	infeasible
pack-rig2-8-lpcc	75	120	46	17	58	0.6	40	0.6
pack-rig2c-16-lpcc	326	525	204	93	123	infeasible	65	infeasible
pack-rig2c-32-lpcc	1580	2725	856	661	765	infeasible	200	infeasible
pack-rig2c-8-lpcc	75	127	46	17	61	0.6	40	0.6
pack-rig2p-16-lpcc	369	565	225	127	294	65.147	166	65.147
pack-rig2p-32-lpcc	1605	2490	961	611	1536	176.47	577	176.47
pack-rig2p-8-lpcc	91	139	49	33	76	1.5269	53	1.5269
pack-rig3-16-lpcc	360	573	204	129	346	0.653857	161	0.653857
pack-rig3-32-lpcc	1490	2342	856	586	746	infeasible	563	infeasible
pack-rig3-8-lpcc	85	139	46	28	65	0.6	42	0.6
pack-rig3c-16-lpcc	360	588	204	129	291	0.737292	130	0.737292
pack-rig3c-32-lpcc	1489	2371	856	585	755	infeasible	317	infeasible
pack-rig3c-8-lpcc	85	146	46	28	64	0.6	35	0.6
portfl-i-1-lpcc	87	99	13	12	62	-0.446267	28	-0.446267
portfl-i-2-lpcc	87	99	13	12	59	-0.502267	26	-0.502267
portfl-i-3-lpcc	87	99	13	12	60	-0.426667	36	-0.426667
portfl-i-4-lpcc	87	99	13	12	61	-0.371267	28	-0.371267
portfl-i-6-lpcc	87	99	13	12	60	-0.295667	32	-0.295667
qpec1-lpcc	30	39	0	20	30	unbounded	1	unbounded
qpec2-lpcc	30	39	0	20	20	unbounded	0	unbounded
qpecgen-100-1-lpcc	105	202	0	100	167	unbounded	7689	(i)
qpecgen-100-2-lpcc	110	202	0	100	197	unbounded	19385	(i)
qpecgen-100-3-lpcc	110	204	0	100	604	unbounded	16161	(i)
qpecgen-100-4-lpcc	120	204	0	100	478	unbounded	23855	(i)
qpecgen-200-1-lpcc	210	404	0	200	1047	unbounded	24243	(i)
qpecgen-200-2-lpcc	220	404	0	200	1838	unbounded	43704	(i)

Table A4. Results of the 168 programs in MacMPEC-LPCC test set, Part IV.

Program	$n$	$m$	$m_e$	$p$	Pivoting Algor.		Filter MPEC	
					# Pivots	Min.	# Pivots	Min.
qpecgen-200-3-lpcc	220	408	0	200	2825	unbounded	41805	-1.06E+17
qpecgen-200-4-lpcc	240	408	0	200	1022	unbounded	27416	-8.34E+16
ralph1-lpcc	2	2	0	1	2	0	5	-6.86E-07
ralph2-lpcc	2	1	0	1	1	unbounded	3	unbounded
ralphmod-lpcc	104	203	0	100	136	-208.88	1515	-502076073
scale1-lpcc	2	1	0	1	0	unbounded	2	unbounded
scale2-lpcc	2	1	0	1	0	unbounded	2	unbounded
scale3-lpcc	2	1	0	1	0	unbounded	3	unbounded
scale4-lpcc	2	1	0	1	0	unbounded	3	unbounded
scale5-lpcc	2	1	0	1	0	unbounded	3	unbounded
scholtes1-lpcc	3	2	0	1	3	unbounded	2	unbounded
scholtes2-lpcc	3	2	0	1	4	-1	2	-1
scholtes3-lpcc	2	1	0	1	1	unbounded	1	-1.00E+20
scholtes4-lpcc	3	4	0	1	4	0	6	-3.42E-07
scholtes5-lpcc	3	3	0	2	1	unbounded	3	unbounded
sl1-lpcc	8	11	2	3	6	-44	10	-44
stackelberg1-lpcc	3	4	1	1	2	-19000	0	-19000
tap-09-lpcc	86	136	32	32	76	86.439	118	86.439
tap-15-lpcc	194	328	68	83	159	139.71	274	139.71
taxmcp-lpcc	12	24	3	10	22	infeasible	134	infeasible
siouxfls-lpcc	2403	4703	628	1748	10800	-23.353	97578	-23.353
siouxfls1-lpcc	2403	4703	628	1748	11781	500.35	12225	500.35
water-net-lpcc	66	116	36	14	37	467.60	11	467.60
water-FL-lpcc	213	373	116	44	135	2119.8	43	2119.8

- [17] Fletcher, R. and Matthews, S. P. J. (1984). Stable modification of explicit  $LU$  factors for simplex updates. *Math. Program.*, 30(3):267–284.
- [18] Fourer, R., Gay, D. M., and Kernighan, B. W. (2002). *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press, 2nd edition.
- [19] Fukushima, M. and Tseng, P. (2002). An implementable active-set algorithm for computing a B-stationary point for a mathematical program with linear complementarity constraints. *SIAM J. Optim.*, 12:724–739.
- [20] Gill, P. E., Murray, W., and Wright, M. H. (1990). *Numerical Linear Algebra and Optimization*. Addison Wesley Publishing Company.
- [21] Goldfarb, D. and Reid, J. K. (1977). A practical steepest-edge simplex algorithm. *Math. Program.*, 12:361–371.
- [22] Hansen, P., Jaumard, B., and Savard, G. (1992). New branch-and-bound rules for linear bilevel programming. *SIAM J. Sci. Stat. Comput.*, 13(5):1194–1217.
- [23] Hoheisel, T. and Kanzow, C. (2009). On the abadie and guignard constraint qualifications for mathematical programmes with vanishing constraints. *Optimization*, 58:431–448.
- [24] Hu, J., Mitchell, J. E., Pang, J.-S., Bennet, K. P., and Kunapuli, G. (2008a). On the global solution of linear programs with linear complementarity constraints. *SIAM J. Optim.*, 19:445–471.
- [25] Hu, J., Mitchell, J. E., Pang, J.-S., Bennett, K. P., and Kunapuli, G. (2008b). On the global solution of linear programs with linear complementarity constraints. *SIAM Journal on Optimization*, 19(1):445–471.
- [26] Hu, X. M. and Ralph, D. (2004). Convergence of a penalty method for mathematical programming with complementarity constraints. *J. Optim. Theory Appl.*, 123:365–390.
- [27] Ibaraki, T. (1971). Complementary programming. *Oper. Res.*, 19:1523–1529.
- [28] Ibaraki, T. (1973). The use of cuts in complementary programming. *Oper. Res.*, 21:353–359.
- [29] Jiang, H. and Ralph, D. (2000). Smooth SQP methods for mathematical programs with nonlinear complementarity constraints. *SIAM J. Optim.*, 10:779–808.
- [30] Jiang, H. and Ralph, D. (2004). Extension of quasi-Newton methods to mathematical programs with complementarity constraints. *Comput. Optim. Appl.*, 123:365–390.
- [31] Júdice, J. J. and Faustino, A. M. (1992). A sequential LCP method for bilevel linear programming. *Ann. Oper. Res.*, 34(1-4):89–106.
- [32] Júdice, J. J., Sherali, H. D., Ribeiro, I. M., and Faustino, A. M. (2006). A complementarity-based partitioning and disjunctive cut algorithm for mathematical programming problems with equilibrium constraints. *J. Global Optim.*, 36(1):89–114.
- [33] Júdice, J. J., Sherali, H. D., Ribeiro, I. M., and Faustino, A. M. (2007). Complementarity active-set algorithm for mathematical programming problems with equilibrium constraints. *J. Optim. Theory Appl.*, 134(3):467–481.
- [34] Leyffer, S. (2000). MacMPEC: AMPL collection of MPECs. Web page, [www.mcs.anl.gov/~leyffer/MacMPEC/](http://www.mcs.anl.gov/~leyffer/MacMPEC/).
- [35] Leyffer, S. (2005). The penalty interior point method fails to converge. *Optim. Methods Soft.*, 20:559–568.
- [36] Leyffer, S. (2006). Complementarity constraints as nonlinear equations: Theory and numerical experience. In Dempe,

- S. and Kalashnikov, V., editors, *Optimization with Multivalued Mappings*, pages 169–208. Springer.
- [37] Leyffer, S., López-Calva, G., and Nocedal, J. (2006). Interior methods for mathematical programs with complementarity constraints. *SIAM J. Optim.*, 17(1):52–77.
- [38] Leyffer, S. and Munson, T. (2007). A globally convergent filter method for MPECs. Preprint ANL/MCS-P1457-0907, Argonne National Laboratory, Mathematics and Computer Science Division.
- [39] Önal, H. (1993). A modified simplex approach for solving bilevel linear programming problems. *European J. Oper. Res.*, 67:126–135.
- [40] Pang, J.-S. and Fukushima, M. (1999). Complementarity constraint qualifications and simplified B-stationarity conditions for mathematical programs with equilibrium constraints. *Comput. Optim. Appl.*, 13(1-3):111–136.
- [41] Scheel, H. and Scholtes, S. (2000). Mathematical program with complementarity constraints: Stationarity, optimality and sensitivity. *Math. of Oper. Res.*, 25:1–22.
- [42] Scholtes, S. (2001). Convergence properties of a regularization scheme for mathematical programs with complementarity constraints. *SIAM J. Optim.*, 11:918–936.
- [43] Stange, P., Griewank, A., and Bollhöfer, M. (2007). On the efficient update of rectangular  $LU$ -factorizations subject to low rank modifications. *Electron. Trans. Numer. Anal.*, 26:161–177.
- [44] Ye, J. J. (1999). Optimality conditions for optimization problems with complementarity constraints. *SIAM J. Optim.*, 9(2):374–387.
- [45] Ye, J. J. (2005). Necessary and sufficient optimality conditions for mathematical programs with equilibrium constraints. *J. Math. Anal. Appl.*, 307:350–369.

The submitted manuscript has been created by the UChicago Argonne, LLC, Operator of Argonne National Laboratory (“Argonne”) under Contract No. DE-AC02-06CH11357 with the U.S. Department of Energy. This work was also supported by NSF grant 0631622. The U.S. Government retains for itself, and others acting on its behalf, a paid-up, nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.