ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

A Pivoting Algorithm for Linear Programs with Complementarity Constraints[1]

Haw-ren Fang, Sven Leyffer, and Todd S. Munson

Mathematics and Computer Science Division

Preprint ANL/MCS-P1680-1009

October 27, 2009

[1]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, and also supported by NSF grant 0631622.

Contents

A Pivoting Algorithm for Linear Programming with Linear Complementarity Constraints*

Haw-ren Fang† Sven Leyffer‡† Todd S. Munson†

October 27, 2009

Abstract

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.

1 Introduction

In the past decade, extensive efforts have been spent on mathematical programming with equilibrium constraints (MPEC). These research efforts center on constraint qualifications and stationarity conditions (Pang and Fukushima, 1999; Scheel and Scholtes, 2000; Ye, 2005, 1999) and algorithms based on nonlinear programming (NLP) techniques to obtain stationary points (Anitescu, 2005; Anitescu et al., 2007; Benson et al., 2006; Facchinei et al., 1999; Fletcher and Leyffer, 2004; Fletcher et al., 2006; Fukushima and Tseng, 2002; Hu and Ralph, 2004; Jiang and Ralph, 2000, 2004; Leyffer, 2005, 2006; Leyffer et al., 2006; Scholtes, 2001).

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 (Leyffer and Munson, 2007), based on sequentially solving a linear model of an MPEC, a linear program with linear complementarity constraints (LPCC). It motivates us to investigate new techniques to solve LPCCs.

LPCC is closely related to bilevel linear programming and mixed-integer programming. 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 include cutting plane (Hu et al., 2008; Ibaraki, 1971, 1973) and branch-and-bound methods (Audet et al., 1997, 2007; Hansen et al., 1992). Both approaches are based on mixed-integer programming. In addition, a penalty method was also proposed by Önal (1993) and analyzed by Campêlo and Scheimberg (2000).

*Preprint ANL/MCS-P1680-1009

†Mathematical & Computer Science Devision, Argonne National Laboratory, 9700 S. Cass Ave., Argonne, IL 60439, USA. Email: {hrfang,leyffer,tmunson}@mcs.anl.gov.

‡To whom correspondence should be addressed.

Rather than transforming the LPCC into another type of program, we consider the LPCC itself. We develop a pivoting algorithm that can handle linear complementarity constraints, based on the classical simplex method for linear programming. Our algorithm 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 (Goldfarb and Reid, 1977; Stange et al., 2007) and low-rank modification of LU factorization for active-set updates (Bartels and Golub, 1969a,b; Fletcher and Matthews, 1984; Stange et al., 2007), can be used in our algorithm to improve its efficiency.

This rest of the 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 Bouligand 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

$$\begin{cases} \underset{x}{\text{minimize}} & g^T x \\ \text{subject to} & a_i^T x \geq b_i, & i = 1,\dots,m \\ & 0 \leq (a_i^T x - b_i) \perp (a_{p+i}^T x - b_{p+i}) \geq 0, & i = m+1,\dots,m+p, \end{cases} \tag{2.1}$$

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 (2.1) 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,

$$c(i) = \begin{cases} \texttt{NULL}, & \text{if } i \leq m; \\ i+p, & \text{if } m+1 \leq i \leq m+p; \\ i-p, & \text{if } m+p+1 \leq i \leq m+2p. \end{cases} \tag{2.2}$$

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

$$a_i^T x \geq b_i, \qquad i = 1,\dots,m+2p. \tag{2.3}$$

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

$$(a_i^T x - b_i)(a_{c(i)}^T x - b_{c(i)}) = 0, \qquad i = m+1,\dots,m+p. \tag{2.4}$$

A point is called *feasible* if it is complementary and linear feasible, that is, satisfying all the constraints in (2.1).

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 (Hoheisel and Kanzow, 2009; Pang and Fukushima, 1999; Scheel and Scholtes, 2000; Ye, 2005, 1999) 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 $a_{c(i)}^T \hat{x} = b_{c(i)}$ (Scholtes, 2001). It is also sometimes called a degenerate or lower-level degenerate complementarity condition. We denote the index set of nonstrict complementarity conditions by

$$\mathcal{D}(\hat{x}) = \left\{ i : a_i^T \hat{x} = b_i^T \wedge a_{c(i)}^T \hat{x} = b_{c(i)},\ i = m+1, \ldots, m+p \right\}. \tag{2.5}$$

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

$$\begin{cases} g - \sum_{i=1}^{m+2p} y_i (a_i^T \hat{x} - b_i) = 0, & \\ 0 \le (a_i^T \hat{x} - b_i) \perp y_i \ge 0, & \forall i \in \{1, \ldots, m\}; \\ a_i^T \hat{x} > b_i \Rightarrow y_i = 0, & \forall i \in \{m+1, \ldots, m+2p\}; \\ y_i \ge 0 \wedge y_{c(i)} \ge 0, & \forall i \in \mathcal{D}(\hat{x}). \end{cases} \tag{2.6}$$

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 $\hat{x}$, then in a neighborhood of $\hat{x}$, the LPCC problem (2.1) is reduced to an LP problem. If $\hat{x}$ also solves this LP, then KKT conditions of this LP are equivalent to the strong stationarity conditions defined in Definition 2.1. Therefore, a strongly stationary point of LPCC (2.1) is a local minimizer, but not vice versa. Next, we review a necessary and sufficient condition for stationarity.

A *Bouligand-stationary or B-stationary* point is a point at which no linearized feasible stationary descend directions exist (Scheel and Scholtes, 2000). An equivalent, more convenient definition is given next.

Definition 2.2 *Given a feasible point $\hat{x}$ of (2.1) 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 complementarity conditions:*

$$\begin{cases} \underset{x}{\text{minimize}} & g^T x & \\ \text{subject to} & a_i^T x \ge b_i, & \forall i \in \{1, \ldots, m\}; \\ & a_i^T x = b_i \text{ and } a_{c(i)}^T x \ge b_{c(i)}, & \forall i \in \{m+1, \ldots, m+p\} \setminus \mathcal{D}(\hat{x}) \text{ and } a_i^T \hat{x} = b_i; \\ & a_i^T x \ge b_i \text{ and } a_{c(i)}^T x = b_{c(i)}, & \forall i \in \{m+1, \ldots, m+p\} \setminus \mathcal{D}(\hat{x}) \text{ and } a_{c(i)}^T \hat{x} = b_{c(i)}; \\ & a_i^T x = b_i \text{ and } a_{c(i)}^T x \ge b_{c(i)}, & \forall i \in \mathcal{P}; \\ & a_i^T x \ge b_i \text{ and } a_{c(i)}^T x = b_{c(i)}, & \forall i \in \mathcal{D}(\hat{x}) \setminus \mathcal{P}. \end{cases} \tag{2.7}$$

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 2.1 *An equivalent statement in terms of multipliers is that for all $\mathcal{P} \subseteq \mathcal{D}(\hat{x})$ there exist multipliers*

$y_1, \ldots, y_{m+2p}$, such that

$$\begin{cases} g - \sum_{i=1}^{m+2p} y_i (a_i^T \hat{x} - b_i) = 0, & \\ 0 \le (a_i^T \hat{x} - b_i) \perp y_i \ge 0, & \forall i \in \{1, \ldots, m\}; \\ a_i^T \hat{x} > b_i \Rightarrow y_i = 0, & \forall i \in \{m+1, \ldots, m+2p\} \setminus \mathcal{D}(\hat{x}); \\ y_{c(i)} \ge 0, & \forall i \in \mathcal{P}; \\ y_i \ge 0, & \forall i \in \mathcal{D}(\hat{x}) \setminus \mathcal{P} \end{cases} \tag{2.8}$$

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 (2000, page 8) give an example where a vertex is B-stationary but not strongly stationary:

$$\begin{cases} \underset{x_1, x_2, x_3}{\text{minimize}} & x_1 + x_2 - x_3 & \\ \text{subject to} & 4x_1 - x_3 \ge 0, & \text{indexed by 1;} \\ & 4x_2 - x_3 \ge 0, & \text{indexed by 2;} \\ & 0 \le x_1 \perp x_2 \ge 0, & \text{indexed by 3 and 4.} \end{cases} \tag{2.9}$$

The only feasible vertex of (2.9) 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

$$\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 4 & 0 & 1 & 0 \\ 0 & 4 & 0 & 1 \\ -1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}. \tag{2.10}$$

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 $y_3 \ge 0$ and $y_4 \ge 0$. Thus, $(0, 0, 0)$ is not strongly stationary.

To prove B-stationarity, we observe that the two LP pieces of (2.9) correspond to $x_1 = 0 \le x_2$ and $x_1 \ge 0 = x_2$. Using (2.10), 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 (2.1) 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 (2.1), we need the following additional condition:

$$\{i, c(i)\} \cap \mathcal{W} \neq \emptyset, \qquad \forall i \in \{m+1, \ldots, m+p\}, \tag{3.1}$$

where $c(i)$, defined in (2.2), is the index of the other complementarity constraint of constraint i. Condition (3.1) 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 $\mathcal{W}_1$:

$$\mathcal{W}_0 = \mathcal{W} \cap \{1, \ldots, m\}, \qquad \mathcal{W}_1 = \mathcal{W} \cap \{m+1, \ldots, m+2p\}, \tag{3.2}$$

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 $j \in \{m+1, \ldots, m+p\}$.

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 (2.1) is

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

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 (Bartels and Golub, 1969a,b; Fletcher and Matthews, 1984; Stange et al., 2007).

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 (2.4). 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 (2.4). Otherwise, $j \in \mathcal{W}_1$ is complementary. Under the nondegeneracy assumption, the direction s_j does not violate (2.4) 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:

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

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

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

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 (2.3). When moving along the direction s_q, an inactive constraint $a_j x \geq b_j$ can become active only if $a_j^T s_q$ less than 0. To be precise, let $\alpha_j \geq 0$ be the step length to satisfy the inequality j not in working set. Thus, $a_j^T(\hat{x} + \alpha_j s_q) - b_j \geq 0$ implies $\alpha_j \leq (b_j - a_j^T \hat{x})/a_j^T s_q$ if $a_j^T s_q < 0$. We conclude that while moving along s_q, the maximal step length $\hat{\alpha}_r$ to satisfy all inequalities is

$$\min \left\{ \frac{b_j - a_j^T \hat{x}}{a_j^T s_q} : a_j^T s_q < 0, j \notin \mathcal{W} \right\}, \tag{3.3}$$

where r is the index minimizer indicating the entering constraint. We call (3.3) 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 ∞, 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.

1: // Given vertex $\hat{x}$ associated with a working set $\mathcal{W}$ satisfying (3.1).
2: Form $A := [a_j]_{j \in \mathcal{W}}$ and $b := [b_j]_{j \in \mathcal{W}}$.
3: **repeat**
4: Compute current vertex $\hat{x} := A^{-T} b$.
5: Compute multipliers $\hat{y} \equiv [\hat{y}_i]_{i \in \mathcal{W}} := A^{-1} g$.
6: Compute $\hat{y}_q := \min \left\{ 0, \hat{y}_i : i \in \mathcal{W}_0 \vee (i \in \mathcal{W}_1 \wedge c(i) \in \mathcal{W}_1) \right\}$.
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: $\hat{\alpha}_r := \min_{\substack{j \notin \mathcal{W} \\ a_j^T s_q < 0}} \left\{ \frac{b_j - a_j^T \hat{x}}{a_j^T s_q}, \infty \right\}$.
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: **end if**
19: **end if**
20: **until** $\hat{x}$ is strongly stationary or $\hat{\alpha}_r = \infty$.

Algorithm 1: A generalized pivoting algorithm for LPCC (2.1).

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 $\hat{y}_q = 0$, we have $\hat{y}^{(0)} \geq 0$, which implies that the first three conditions of (2.6) are satisfied. For the last condition, $\hat{y}^{(1)} \geq 0$ 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 Scheel and Scholtes (2000, Theorem 2). ∎

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

$$\left\{\begin{array}{lll} \underset{x_1,x_2,x_3,x_4,x_5}{\text{minimize}} & 4x_1 - 2x_2 + x_3 - x_5 & \\ \text{subject to} & x_1 \geq 0,\ x_2 \geq 0, & \text{indexed by 1,2;} \\ & x_1 + 2x_4 \geq 2, & \text{indexed by 3;} \\ & x_3 - x_4 - x_5 \geq -2, & \text{indexed by 4;} \\ & 0 \leq x_3 \perp x_1 - x_2 + x_3 + 1 \geq 0, & \text{indexed by 5 and 8;} \\ & 0 \leq x_4 \perp x_1 - x_3 + 2 \geq 0, & \text{indexed by 6 and 9;} \\ & 0 \leq x_5 \perp x_3 - x_4 + 1 \geq 0, & \text{indexed by 7 and 10.} \end{array}\right. \tag{3.4}$$

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 (3.3) 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 (3.3) 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 (2.4) 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

$$\begin{cases} \underset{x_1, x_2, x_3}{\text{minimize}} & -x_1 & \\ \text{subject to} & x_1 - x_2 + x_3 \geq 0, & \text{indexed by 1;} \\ & x_1 + x_2 + x_3 \geq 0, & \text{indexed by 2;} \\ & -x_1 \geq -1, & \text{indexed by 3;} \\ & 0 \leq x_1 \perp x_2 \geq 0, & \text{indexed by 4 and 5.} \end{cases} \tag{4.1}$$

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

$$A = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}, \qquad g = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}, \qquad \hat{y} = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}. \tag{4.2}$$

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:

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

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 (2.4). Thus, condition (3.1) is relaxed to

$$\{i, c(i)\} \cap \bar{\mathcal{W}} \neq \emptyset \quad \text{for} \quad i = m+1, \ldots, m+p. \tag{4.3}$$

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 violates any constraint $r \in \mathcal{E}$, we move r from $\mathcal{E}$ to $\mathcal{W}$ right away. Otherwise, we proceed with the usual ratio test (3.3) 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 (4.3) 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 17 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.

3. $A = [a_j]_{j \in \mathcal{W}}$ is always guaranteed to be nonsingular. The reason is that no matter which entering constraint r is chosen in line 17 of Algorithm 2 or in the ratio test in line 20, 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 (2.1), 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,

   $$\forall j \in \mathcal{E}, \quad c(j) \notin \bar{\mathcal{W}}, \tag{4.4}$$

   to exclude unnecessary complementarity constraints from $\mathcal{E}$ and therefore potentially reduce the number of pivoting steps. If the initial extended working set satisfies (4.4), then (4.4) remains satisfied in lines 37–39.

Now we apply Algorithm 2 to the LPCC (4.1). 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 (4.3) is satisfied. From (4.2) we conclude that the leaving constraint q is 3, 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 $c(q) = 4 \notin \mathcal{W} \cup \mathcal{E}$, we add it to $\mathcal{E}$ and have $\mathcal{E} = \{5\}$. At the test in lines 13–15, constraint 5 from $\mathcal{E}$ has

// Given vertex $\hat{x}$ associated with an extended working set $\bar{\mathcal{W}} = \mathcal{W} \cup \mathcal{E}$ satisfying (4.3).
// All constraints in $\bar{\mathcal{W}}$ are active at $\hat{x}$, and $\mathcal{W}$ consists of n linearly independent constraints.
$\mathcal{A}_1 := \emptyset$; $\mathcal{A}_2 := \emptyset$.
Form $A := [a_j]_{j \in \mathcal{W}}$ and $b := [b_j]_{j \in \mathcal{W}}$.
repeat
    Compute current vertex $\hat{x} := A^{-T} b$.
    Compute multipliers $\hat{y} \equiv [\hat{y}_i]_{i \in \mathcal{W}} := A^{-1} g$.
    Compute $\hat{y}_q := \min \left\{ 0, \hat{y}_i : i \in \mathcal{W}_0 \vee (i \in \mathcal{W}_1 \wedge c(i) \in \mathcal{W}_1) \right\}$.
    // The index q indicates the constraint to quit the working set.
    if $\hat{y}_q = 0$ **then**
        return: $\hat{x}$ is strongly stationary.
    else
        if $q \in \mathcal{W}_1$ and $c(q) \notin \bar{\mathcal{W}}$ **then**
            $\mathcal{E} := \mathcal{E} \cup \{c(q)\}$
        end if
        Compute search direction s_q as the column of A^{-T} corresponding to $\hat{y}_q$.
        if $\exists r \in \mathcal{E}$ such that $a_r^T s_q \neq 0$ **then**
            $\mathcal{E} := \mathcal{E} \setminus \{r\}$
        else // The index r indicates the constraint to enter the working set.
            Ratio test: $\hat{\alpha}_r := \min\limits_{\substack{j \notin \bar{\mathcal{W}} \\ a_j^T s_q < 0}} \left\{ \dfrac{b_j - a_j^T \hat{x}}{a_j^T s_q}, \infty \right\}$.
            if $\hat{\alpha}_r = \infty$ **then**
                return: the LPCC is unbounded.
            end if
        end if
        (`cycle`, $\mathcal{A}_1, \mathcal{A}_2$) =DetectCycle($q, r, \hat{\alpha}_r, \mathcal{A}_1, \mathcal{A}_2$);
        if `cycle` = `true` **then**
            (`status`, $\mathcal{W}, \mathcal{E}$) = AntiCycle($\mathcal{W}, \mathcal{E}$);
            if `status` = `B_stationary` **then**
                return: $\hat{x}$ is B-stationary.
            else if `status` = `unbounded` **then**
                return: the LPCC is unbounded.
            else // `status` = `cycle_breaks`
                Update $A := [a_j]_{j \in \mathcal{W}}$, and $b := [b_j]_{j \in \mathcal{W}}$.
            end if
        else
            Update $\mathcal{W} := \mathcal{W} \cup \{r\} \setminus \{q\}$, $A := [a_j]_{j \in \mathcal{W}}$, and $b := [b_j]_{j \in \mathcal{W}}$.
            if $c(r) \in \mathcal{E}$ **then**
                $\mathcal{E} := \mathcal{E} \setminus \{c(r)\}$
            end if
        end if
    end if
until $\hat{x}$ is strongly stationary or B-stationary, or $\hat{\alpha}_r = \infty$.

Algorithm 2: A revised pivoting algorithm for LPCC (2.1).

gradient $a_4 = (0, 1, 0)$. Since $a_4^T s_q = 0$, it cannot be the entering constraint. The entering constraint is $-x_1 \geq -1$, determined by ratio test in line 20. 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 (2.9) 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 (Chvátal, 1983, Theorem 3.3); (Gill et al., 1990, 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.

$$\begin{cases} \underset{x_1,x_2,x_3}{\text{minimize}} & x_1 + x_2 + x_3 + x_4 - x_5 - x_6 & \\ \text{subject to} & 4x_1 - x_5 \geq 0, & \text{indexed by 1;} \\ & 4x_2 - x_5 \geq 0, & \text{indexed by 2;} \\ & 4x_3 - x_6 \geq 0, & \text{indexed by 3;} \\ & 4x_4 - x_6 \geq 0, & \text{indexed by 4;} \\ & 0 \leq x_1 \perp x_2 \geq 0, & \text{indexed by 5 and 6;} \\ & 0 \leq x_3 \perp x_4 \geq 0, & \text{indexed by 7 and 8.} \end{cases} \tag{5.1}$$

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 (1983). 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 $\mathrm{LP}(\hat{x}, \mathcal{P})$ for all possible $\mathcal{P} \subseteq \mathcal{D}(\hat{x})$. We apply an anticycling rule, such as Bland's least index rule, to each $\mathrm{LP}(\hat{x}, \mathcal{P})$. Then it follows that $\hat{x}$ is B-stationary if and only if $\hat{x}$ is a minimizer to $\mathrm{LP}(\hat{x}, \mathcal{P})$ 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 $\mathrm{LP}(\hat{x}, \mathcal{P})$ for some $\mathcal{P} \subseteq \mathcal{D}(\hat{x})$, assume that we obtain a set of multipliers $\hat{y}_i$ that satisfies (2.8) and therefore

```
1: function [cycle, A_1, A_2] = CycleDetect(q, r, α̂_r, A_1, A_2)
2:     // A_1 and A_2 are arrays containing the recent leaving and entering constraints.
3:     // q and r are the leaving and entering constraints; α̂_r is the step length.
4:     if α̂_r > 0 then
5:         // The objective of (2.1) is reduced, so previous steps cannot trigger a cycling.
6:         cycle := false
7:     else
8:         Update l := size(A_1), A_1(l) := q, and A_2(l) := r.
9:         if there exists k such that A_1(l−k+1, ..., l) and A_2(l−k+1, ..., l) form the same set then
10:            // Cycling is detected.
11:            cycle := true
12:        else
13:            cycle := false
14:        end if
15:    end if
16: end function
```

Algorithm 3: Cycle detection for Algorithm 2.

$\hat{x}$ is a solution to 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 LP$(\hat{x}, \mathcal{P} \setminus \{i\})$. In general, given a set of multipliers satisfying (2.8) at $\hat{x}$, we let

$$\mathcal{R}_1 = \{i : \hat{y}_i \geq 0 \wedge i \in \mathcal{P}\}, \qquad \mathcal{R}_2 = \{i : \hat{y}_{c(i)} \geq 0 \wedge i \in \mathcal{D}(\hat{x}) \setminus \mathcal{P}\}, \tag{5.2}$$

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

$$\text{LP}(\hat{x}, \mathcal{P} \cup \mathcal{S}_2 \setminus \mathcal{S}_1) \quad \text{for all} \quad \mathcal{S}_1 \subseteq \mathcal{R}_1, \quad \mathcal{S}_2 \subseteq \mathcal{R}_2. \tag{5.3}$$

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

$$\mathcal{P} \cup \mathcal{P}^c \subseteq \bar{\mathcal{W}}, \qquad \mathcal{P}^c = \{c(i) : i \in \mathcal{D}(\hat{x}) \setminus \mathcal{P}\}. \tag{5.4}$$

function [status, $\mathcal{W}$, $\mathcal{E}$] = ANTICYCLE($\mathcal{W}$, $\mathcal{E}$)
  // The current vertex is $\hat{x}$, associated with an extended working set $\bar{\mathcal{W}} = \mathcal{W} \cup \mathcal{E}$.
  Set $\mathcal{U} := 2^{\mathcal{D}(\hat{x})}$, where $\mathcal{D}(\hat{x}) = \{i : a_i^T \hat{x} = b_i \wedge a_{c(i)}^T \hat{x} = b_{c(i)}\}$.
  repeat
    Select $\mathcal{P} \in \mathcal{U}$; let $\mathcal{P}^c := \{c(i) : i \in \mathcal{D}(\hat{x}) \setminus \mathcal{P}\}$.
    // We consider LP($\hat{x}, \mathcal{P}$) and apply Bland's least index rule.
    $\mathcal{E} := \mathcal{E} \cup \{i : i \in \mathcal{P} \cup \mathcal{P}^c \wedge i \notin \bar{\mathcal{W}}\}$
    repeat
      Compute $\hat{y} \equiv [\hat{y}_i]_{i \in \mathcal{W}} := A^{-1} g$, where $A := [a_j]_{j \in \mathcal{W}}$.
      Compute $q := \min\{i : i \in C_0 \cup C_1\}$, where
        $C_0 := \min\{i : i \in \mathcal{W}_0 \wedge \hat{y}_i < 0\}$
        $C_1 := \min\{i : i \in \mathcal{W}_1 \setminus (\mathcal{P} \cup \mathcal{P}^c) \wedge \hat{y}_i < 0\}$
      // Here $\mathcal{W}_0 = \mathcal{W} \cap \{1, \ldots, m\}$ and $\mathcal{W}_1 = \mathcal{W} \cap \{m+1, \ldots, m+2p\}$.
      if $q \neq$ NULL **then**
        $\hat{y}_q < 0$ is the qualified multiplier with smallest index.
        Compute search direction s_q as the column of A^{-T} corresponding to $\hat{y}_q$.
        if $\exists r \in \mathcal{E}$ such that $a_r^T s_q \neq 0$ **then**
          $\mathcal{E} := \mathcal{E} \setminus \{r\}$
        else
          Ratio test: $\hat{\alpha}_r := \min\limits_{\substack{j \notin \bar{\mathcal{W}} \\ a_j^T s_q < 0}} \left\{ \dfrac{b_j - a_j^T \hat{x}}{a_j^T s_q}, \infty \right\}$.
          If multiple choices of r exist, choose the smallest one.
          if $\hat{\alpha}_r = \infty$ **then**
            // LP($\hat{x}, \mathcal{P}$) is unbounded, so is the LPCC.
            return: status := unbounded.
          end if
          Update $\mathcal{W} := \mathcal{W} \cup \{r\} \setminus \{q\}$.
          if $\hat{\alpha}_r > 0$ **then**, // cycle breaks.
            return: status := cycle_breaks.
          end if
        end if
      end if
    until $q =$ NULL (i.e., no qualified leaving constraint).
    // $\hat{x}$ is a solution to LP($\hat{x}, \mathcal{P}$).
    Set $\mathcal{R}_1 := \{i : \hat{y}_i \geq 0 \wedge i \in \mathcal{P}\}$, $\mathcal{R}_2 := \{i : \hat{y}_{c(i)} \geq 0 \wedge i \in \mathcal{D}(\hat{x}) \setminus \mathcal{P}\}$
    Update $\mathcal{U} := \mathcal{U} \setminus \{\mathcal{P} \cup \mathcal{S}_2 \setminus \mathcal{S}_1 : \mathcal{S}_1 \subseteq \mathcal{R}_1 \wedge \mathcal{S}_2 \subseteq \mathcal{R}_2\}$
  until $\mathcal{U} = \emptyset$.
  return: status := B_stationary.
end function

Algorithm 4: Anticycling for Algorithm 2.

We can ensure (5.4) 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 (5.4).
- 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 (Chvátal, 1983, Theorem 3.3); (Gill et al., 1990, 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 (2.1).
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 (4.4) may no longer hold because we augment $\mathcal{E}$. Nevertheless, (4.4) is not required by Algorithm 2 to work properly, but it potentially avoids unnecessary pivoting steps. Indeed, we can impose
   $$\mathcal{E} := \left\{ i : i \in \mathcal{E} \wedge c(i) \notin \mathcal{W} \wedge (i \le m+p \vee c(i) \notin \mathcal{E}) \right\}$$
   when returning to Algorithm 2, if (4.4) 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 (2.9) which has only one vertex $\hat{x} = (0, 0, 0)$.

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 LP($\hat{x}, \emptyset$) and cannot leave $\mathcal{W}$. Therefore, $\hat{x}$ is a minimizer of 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 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 LP($\hat{x}, \{3\}$), so it cannot leave the working set $\mathcal{W}$. Hence $\hat{x} = (0, 0, 0)$ is also a minimizer of 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 (5.4). As discussed in Section 4, our $\bar{\mathcal{W}}$ in Algorithm 2 satisfies (4.3), so there exists $\mathcal{P}$ satisfying (5.4). 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,

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

6 Obtaining an Initial LPCC Feasible Vertex

Our pivoting algorithm to solve an LPCC (2.1) 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 (2.1).

Our Phase I is identical to the Phase I of LP, where we find a linear feasible vertex satisfying all the inequalities (2.3). If such a vertex is found, we continue with Phase II to resolve complementary violations in order to satisfy (2.4). 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 (2.1) 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:

$$\begin{aligned} \mathcal{I} &= \{i : (a_i^T \hat{x} - b_i)(a_{c(i)}^T \hat{x} - b_{c(i)}) = 0, i = m+1, \ldots, m+p\}, \\ \mathcal{J} &= \{i : (a_i^T \hat{x} - b_i)(a_{c(i)}^T \hat{x} - b_{c(i)}) > 0, i = m+1, \ldots, m+p\}. \end{aligned} \tag{6.1}$$

We resolve the complementary violation one at a time, and update (6.1) at each iteration. The pseudo-code is given in Algorithm 5.

1: // Given a linearly feasible vertex $\hat{x}$ of LPCC (2.1) from Phase I.
2: Determine $\mathcal{I}$ and $\mathcal{J}$ by (6.1).
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:

$$\begin{cases} \underset{x}{\text{minimize}} & a_j^T x \\ \text{subject to} & a_i^T x \geq b_i, \quad i = 1, \ldots, m+2p, \\ & 0 \leq (a_i^T x - b_i) \perp (a_{c(i)}^T x - b_{c(i)}) \geq 0, \quad i \in \mathcal{I}. \end{cases} \tag{6.2}$$

6: **if** the minimizer x^* of (6.2) satisfies $a_j^T x^* = b_j$ **then**
7: $\mathcal{K} := \emptyset$; $\hat{x} := x^*$; update $\mathcal{I}$ and $\mathcal{J}$ by (6.1).
8: **else**
9: Starting from $\hat{x}$, solve the LPCC:

$$\begin{cases} \underset{x}{\text{minimize}} & a_{c(j)}^T x \\ \text{subject to} & a_i^T x \geq b_i, \quad i = 1, \ldots, m+2p, \\ & 0 \leq (a_i^T x - b_i) \perp (a_{c(i)}^T x - b_{c(i)}) \geq 0, \quad i \in \mathcal{I}. \end{cases} \tag{6.3}$$

10: **if** the minimizer x^* of (6.3) satisfies $a_{c(j)}^T x^* = b_{c(j)}$ **then**
11: $\mathcal{K} := \emptyset$; $\hat{x} := x^*$; update $\mathcal{I}$ and $\mathcal{J}$ by (6.1).
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 (2.1).

We note that we can solve LPCCs (6.2) and (6.3) 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 (6.2) or (6.3).

2. Since we have at least one more satisfied complementarity condition, $\mathcal{I}$ is augmented. After augmentation of $\mathcal{I}$, the condition (4.3) 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 (4.3), 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 (4.3). Finally, the augmentation of $\mathcal{E}$ does not violate of (4.4).

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 (2.1) 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 (2.1) *locally infeasible*.

For every $j \in \mathcal{J}$, we can either solve (6.3) first or solve (6.2) 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 (6.2) or (6.3) 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, continuing to solve LPCCs as presented in Algorithm 5 sometimes helps. 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 (3.4). 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 complementary 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 (6.2) reads as

$$\left\{\begin{array}{lll} \underset{x_1,x_2,x_3,x_4,x_5}{\text{minimize}} & x_4 & \\ \text{subject to} & x_1, x_2 \geq 0, & \text{indexed by 1,2;} \\ & x_1 + 2x_4 \geq 2, & \text{indexed by 3;} \\ & x_3 - x_4 - x_5 \geq -2, & \text{indexed by 4;} \\ & 0 \leq x_3 \perp x_1 - x_2 + x_3 + 1 \geq 0, & \text{indexed by 5 and 8;} \\ & 0 \leq x_4, \quad x_1 - x_3 + 2 \geq 0, & \text{indexed by 6 and 9;} \\ & 0 \leq x_5 \perp x_3 - x_4 + 1 \geq 0, & \text{indexed by 7 and 10.} \end{array}\right. \tag{6.4}$$

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 (6.4), 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 (3.3) 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.

7 Numerical Experiments

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 (Fletcher et al., 2006) that solves MPEC by introducing slack variables s and replacing complementarity conditions $y \perp s$ by $y^T s \le 0$. 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 (Leyffer, 2000), a collection of 168 MPEC test problems written in AMPL (Fourer et al., 2002). AMPL allows more general constraints, including range constraints and equations. AMPL also allows mixed complementarity constraints. See Ferris et al. (1999) 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 `gnash1`j`-lpcc` for $j = 0, 1, \ldots, 9$.

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 (2.5). Recall that Algorithm 4 for anticycling uses a set $\mathcal{U}$ to track the determination of optimality of LP pieces $\mathrm{LP}(\hat{x}, \mathcal{P})$ 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 $|\mathcal{D}(\hat{x})| \le 16$ 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 degree nonstrictness reached	2	max number iterations 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 2–5.

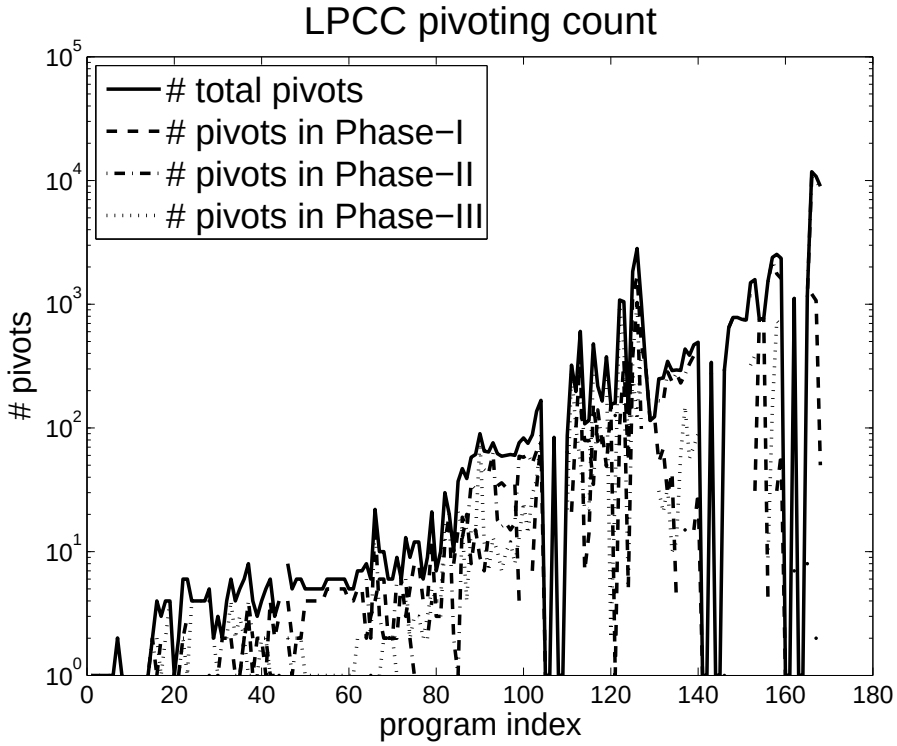

Figure 1: Pivoting counts of our pivoting algorithm using the MacMPEC-LPCC test set.

Figure 2 shows the performance profiles (Dolan and Moré, 2002; Dolan et al., 2006) of our pivoting algorithm and filter MPEC. In the left plot we use all 168 programs 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.

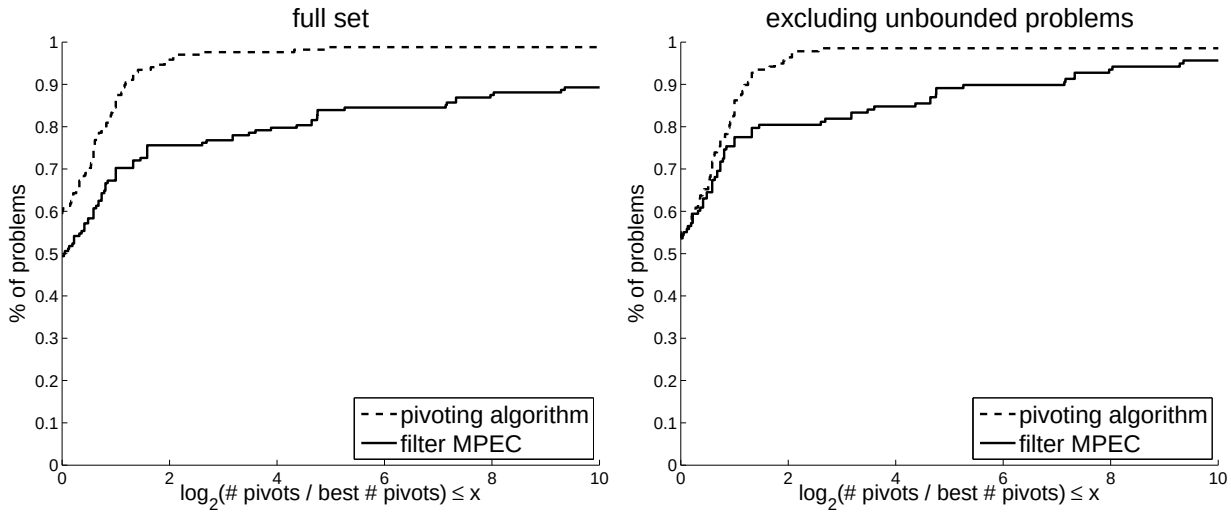

Figure 2: Performance profiles ($\log_2$ scale) using the MacMPEC-LPCC test set.

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.

A MacMPEC-LPCC Program Characteristics

Tables 2–5 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

Anitescu, M. (2005). On using the elastic mode in nonlinear programming approaches to mathematical programs with complementarity constraints. *SIAM J. Optim.*, 15:1203–1236.

Table 2: Results of the 168 programs in the MacMPEC-LPCC test set, Part I.

Program	n	m	m_e	p	# Pivots	
					Pivoting Algor.	Filter MPEC
bard1-lpcc	5	9	1	3	6	8
bard1m-lpcc	6	10	1	3	5	8
bard2-lpcc	12	21	5	3	7	8
bard2m-lpcc	12	21	5	3	7	9
bar-truss-3-lpcc	35	45	28	6	10	15
bilevel1-lpcc	10	17	2	6	8	8
bilevel1m-lpcc	8	13	2	4	6	12
bilevel2-lpcc	16	29	4	8	12	20
bilevel2m-lpcc	16	29	4	8	12	20
bilevel3-lpcc	10	14	6	2	5	3
bilin-lpcc	8	15	0	6	8	165
bem-milanc30-s-lpcc	3436	4901	1968	1464	8985	1517
dempe-lpcc	3	3	1	1	2	3(i)
design-cent-1-lpcc	12	15	6	3	8	97
design-cent-21-lpcc	13	19	6	3	10	9
design-cent-2-lpcc	13	19	6	3	10	10
design-cent-31-lpcc	15	15	6	3	13	65(i)
design-cent-3-lpcc	15	15	6	3	9	63(i)
design-cent-4-lpcc	22	33	10	8	7	7
desilva-lpcc	6	8	2	2	2	0
df1-lpcc	2	2	0	1	1	0
ex9.1.1-lpcc	13	18	7	5	6	5
ex9.1.2-lpcc	8	13	5	2	4	2
ex9.1.3-lpcc	23	35	15	6	10	8
ex9.1.4-lpcc	8	13	5	2	3	3
ex9.1.5-lpcc	13	20	7	5	6	6
ex9.1.6-lpcc	14	21	7	6	9	8
ex9.1.7-lpcc	17	26	9	6	9	7
ex9.1.8-lpcc	11	17	5	3	5	4
ex9.1.9-lpcc	12	18	6	5	6	7
ex9.1.10-lpcc	11	17	5	3	5	4
ex9.2.1-lpcc	10	15	5	4	6	7
ex9.2.2-lpcc	9	14	4	3	3	6
ex9.2.3-lpcc	14	23	8	4	6	6
ex9.2.4-lpcc	8	12	5	2	4	4
ex9.2.5-lpcc	8	11	4	3	5	7
ex9.2.6-lpcc	16	22	6	6	6	4
ex9.2.7-lpcc	10	15	5	4	6	7
ex9.2.8-lpcc	6	9	3	2	3	4
ex9.2.9-lpcc	9	14	5	3	4	2
flp2-lpcc	4	5	0	2	6	8
flp4-1-lpcc	80	90	0	30	90	2829(t)

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

Program	n	m	m_e	p	# Pivots	
					Pivoting Algor.	Filter MPEC
flp4-2-lpcc	110	170	0	60	322	211
flp4-3-lpcc	140	240	0	70	376	970(t)
flp4-4-lpcc	200	350	0	100	1081	548
gauvin-lpcc	3	4	0	2	4	5
gnash10m-lpcc	10	15	5	4	5	9
gnash11m-lpcc	10	15	5	4	5	8
gnash12m-lpcc	10	15	5	4	5	2
gnash13m-lpcc	10	15	5	4	5	2
gnash14m-lpcc	10	15	5	4	5	2
gnash15m-lpcc	10	15	5	4	6	9
gnash16m-lpcc	10	15	5	4	6	9
gnash17m-lpcc	10	15	5	4	6	4
gnash18m-lpcc	10	15	5	4	6	4
gnash19m-lpcc	10	15	5	4	6	4
hakonsen-lpcc	9	17	3	4	0	0
hs044-i-lpcc	20	30	4	10	21	23
incid-set1-8-lpcc	100	170	49	32	75	62
incid-set1-16-lpcc	371	637	225	111	292	205
incid-set1-32-lpcc	1517	2559	961	489	1492	935
incid-set1c-8-lpcc	100	177	49	32	88	78
incid-set1c-16-lpcc	371	652	225	111	435	237
incid-set1c-32-lpcc	1517	2590	961	489	1583	824
incid-set2-8-lpcc	112	177	49	44	108	975
incid-set2-16-lpcc	450	681	225	190	471	18038
incid-set2-32-lpcc	1857	2767	961	829	2532	101792(t)
incid-set2c-8-lpcc	112	184	49	44	115	742
incid-set2c-16-lpcc	450	696	225	190	493	13251
incid-set2c-32-lpcc	1857	2798	961	829	2349	28910(t)
jr1-lpcc	2	1	0	1	1	0
jr2-lpcc	2	1	0	1	1	0
kth1-lpcc	2	1	0	1	1	0
kth2-lpcc	2	1	0	1	1	1
kth3-lpcc	2	1	0	1	1	1
liswet1-050-lpcc	152	203	52	50	155	68(i)
liswet1-100-lpcc	302	403	102	100	309	140(i)
liswet1-200-lpcc	602	803	202	200	647	360(i)
monteiroB-lpcc	131	226	57	57	218(d)	298
monteiro-lpcc	131	226	57	57	165(d)	136
nash1a-lpcc	6	8	2	2	2	5
nash1b-lpcc	6	8	2	2	4	10
nash1c-lpcc	6	8	2	2	6	12
nash1d-lpcc	6	8	2	2	4	11

Table 4: Results of the 168 programs in the MacMPEC-LPCC test set, Part III.

Program	n	m	m_e	p	# Pivots	
					Pivoting Algor.	Filter MPEC
nash1e-lpcc	6	8	2	2	5	7
outrata31-lpcc	5	8	0	4	4	7
outrata32-lpcc	5	8	0	4	4	7
outrata33-lpcc	5	8	0	4	4	7
outrata34-lpcc	5	8	0	4	4	795(i)
pack-comp1-8-lpcc	107	179	49	49	1	27
pack-comp1-16-lpcc	467	753	225	225	1	143
pack-comp1-32-lpcc	1955	3101	961	961	1	624
pack-comp1c-8-lpcc	107	186	49	49	1	27
pack-comp1c-16-lpcc	467	768	225	225	1	141
pack-comp1c-32-lpcc	1955	3132	961	961	1	655
pack-comp1p-8-lpcc	107	164	49	49	84	83
pack-comp1p-16-lpcc	467	708	225	225	339	347
pack-comp1p-32-lpcc	1955	2948	961	961	1115	1049
pack-comp2-8-lpcc	107	179	49	49	1	25
pack-comp2-16-lpcc	467	753	225	225	1	161
pack-comp2-32-lpcc	1955	3101	961	961	1	262
pack-comp2c-8-lpcc	107	186	49	49	1	25
pack-comp2c-16-lpcc	467	768	225	225	1	161
pack-comp2c-32-lpcc	1955	3132	961	961	1	251
pack-comp2p-8-lpcc	107	164	49	49	86	93
pack-comp2p-16-lpcc	467	708	225	225	294	293
pack-comp2p-32-lpcc	1955	2948	961	961	1159	1003
pack-rig1-8-lpcc	70	109	46	9	47	32
pack-rig1-16-lpcc	333	511	204	82	250	185
pack-rig1-32-lpcc	1433	2171	856	505	781	186
pack-rig1c-8-lpcc	70	116	46	9	39	26
pack-rig1c-16-lpcc	333	526	204	82	253	174
pack-rig1c-32-lpcc	1433	2202	856	505	781	186
pack-rig1p-8-lpcc	92	138	49	34	83	72
pack-rig1p-16-lpcc	389	580	225	147	386	240
pack-rig1p-32-lpcc	1711	2571	961	717	2387	7443(t)
pack-rig2-8-lpcc	75	120	46	17	58	40
pack-rig2-16-lpcc	326	510	204	93	115	65
pack-rig2-32-lpcc	1580	2694	856	661	756	201
pack-rig2c-8-lpcc	75	127	46	17	61	40
pack-rig2c-16-lpcc	326	525	204	93	123	65
pack-rig2c-32-lpcc	1580	2725	856	661	765	200
pack-rig2p-8-lpcc	91	139	49	33	76	53
pack-rig2p-16-lpcc	369	565	225	127	294	166
pack-rig2p-32-lpcc	1605	2490	961	611	1536	577
pack-rig3-8-lpcc	85	139	46	28	65	42

Table 5: Results of the 168 programs in the MacMPEC-LPCC test set, Part IV.

Program	n	m	m_e	p	# Pivots	
					Pivoting Algor.	Filter MPEC
pack-rig3-16-lpcc	360	573	204	129	346	161
pack-rig3-32-lpcc	1490	2342	856	586	746	563
pack-rig3c-8-lpcc	85	146	46	28	64	35
pack-rig3c-16-lpcc	360	588	204	129	291	130
pack-rig3c-32-lpcc	1489	2371	856	585	755	317
portfl-i-1-lpcc	87	99	13	12	62	28
portfl-i-2-lpcc	87	99	13	12	59	26
portfl-i-3-lpcc	87	99	13	12	60	36
portfl-i-4-lpcc	87	99	13	12	61	28
portfl-i-6-lpcc	87	99	13	12	60	32
qpec1-lpcc	30	39	0	20	30	1
qpec2-lpcc	30	39	0	20	20	0
qpecgen-100-1-lpcc	105	202	0	100	167	7689(i)
qpecgen-100-2-lpcc	110	202	0	100	197	19385(i)
qpecgen-100-3-lpcc	110	204	0	100	604	16161(i)
qpecgen-100-4-lpcc	120	204	0	100	478	23855(i)
qpecgen-200-1-lpcc	210	404	0	200	1047	24243(i)
qpecgen-200-2-lpcc	220	404	0	200	1838	43704(i)
qpecgen-200-3-lpcc	220	408	0	200	2825	41805
qpecgen-200-4-lpcc	240	408	0	200	1022	27416
ralph1-lpcc	2	2	0	1	2	5
ralph2-lpcc	2	1	0	1	1	3
ralphmod-lpcc	104	203	0	100	136	1515
scale1-lpcc	2	1	0	1	0	2
scale2-lpcc	2	1	0	1	0	2
scale3-lpcc	2	1	0	1	0	3
scale4-lpcc	2	1	0	1	0	3
scale5-lpcc	2	1	0	1	0	3
scholtes1-lpcc	3	2	0	1	3	2
scholtes2-lpcc	3	2	0	1	4	2
scholtes3-lpcc	2	1	0	1	1	1
scholtes4-lpcc	3	4	0	1	4	6
scholtes5-lpcc	3	3	0	2	1	3
siouxfls-lpcc	2403	4703	628	1748	10800	97578
siouxfls1-lpcc	2403	4703	628	1748	11781	12225
sl1-lpcc	8	11	2	3	6	10
stackelberg1-lpcc	3	4	1	1	2	0
tap-09-lpcc	86	136	32	32	76	118
tap-15-lpcc	194	328	68	83	159	274
taxmcp-lpcc	12	24	3	10	22	134
water-net-lpcc	66	116	36	14	37	11
water-FL-lpcc	213	373	116	44	135	43

Anitescu, M., Tseng, P., and Wright, S. J. (2007). Elastic-mode algorithms for mathematical programs with equilibrium constraints. *Math. Program.*, 110:337–371.

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.

Audet, C., Savard, S., and Zghal, W. (2007). New branch-and-cut algorithm for bilevel linear programming. *J. Optim. Theory Appl.*, 134:353–370.

Bartels, R. H. and Golub, G. H. (1969a). Algorithm 350: Simplex method procedure employing *LU* decomposition. *Commun. of ACM*, 12:275–281.

Bartels, R. H. and Golub, G. H. (1969b). The simplex method of linear programming using *LU* decomposition. *Commun. of ACM*, 12:266–268.

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.

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.

Chvátal, V. (1983). *Linear Programming*. W. H. Freeman & Company.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Math. Program. Ser. A*, 91:201–213.

Dolan, E. D., Moré, J. J., and Munson, T. S. (2006). Optimality measures for performance profiles. *SIAM J. Optim.*, 16(3):891–909.

Facchinei, F., Jiang, H., and Qi, L. (1999). A smoothing method for mathematical programs with equilibrium constraints. *Math. Program.*, 85:107–134.

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.

Fletcher, R. and Leyffer, S. (2004). Solving mathematical program with complementarity constraints as nonlinear programs. *Optim. Methods Soft.*, 19(1):15–40.

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.

Fletcher, R. and Matthews, S. P. J. (1984). Stable modification of explicit *LU* factors for simplex updates. *Math. Program.*, 30(3):267–284.

Fourer, R., Gay, D. M., and Kernighan, B. W. (2002). *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press, 2nd edition.

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.

Gill, P. E., Murray, W., and Wright, M. H. (1990). *Numerical Linear Algebra and Optimization*. Addison Wesley Publishing Company.

Goldfarb, D. and Reid, J. K. (1977). A practical steepest-edge simplex algorithm. *Math. Program.*, 12:361–371.

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.

Hoheisel, T. and Kanzow, C. (2009). On the abadie and guignard constraint qualifications for mathematical programmes with vanishing constraints. *Optimization*, 58:431–448.

Hu, J., Mitchell, J. E., Pang, J.-S., Bennet, K. P., and Kunapuli, G. (2008). On the global solution of linear programs with linear commplementarity constraints. *SIAM J. Optim.*, 19:445–471.

Hu, X. M. and Ralph, D. (2004). Convergence of a penality method for mathematical programming with complementarity constraints. *J. Optim. Theory Appl.*, 123:365–390.

Ibaraki, T. (1971). Complementary programming. *Oper. Res.*, 19:1523–1529.

Ibaraki, T. (1973). The use of cuts in complementary programming. *Oper. Res.*, 21:353–359.

Jiang, H. and Ralph, D. (2000). Smooth SQP methods for mathematical programs with nonlinear complementarity constraints. *SIAM J. Optim.*, 10:779–808.

Jiang, H. and Ralph, D. (2004). Extension of quasi-Newton methods to mathematical programs with complementarity constraints. *Comput. Optim. Appl.*, 123:365–390.

Leyffer, S. (2000). MacMPEC: AMPL collection of MPECs. Web page, `www.mcs.anl.gov/~leyffer/MacMPEC/`.

Leyffer, S. (2005). The penalty interior point method fails to converge. *Optim. Methods Soft.*, 20:559–568.

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.

Leyffer, S., Lopéz-Calva, G., and Nocedal, J. (2006). Interior methods for mathematical programs with complementarity constraints. *SIAM J. Optim.*, 17(1):52–77.

Leyffer, S. and Munson, T. S. (2007). A globally convergent filter method for MPECs. Preprint ANL/MCS-P1457-0907, Argonne National Laboratory, Mathematics and Computer Science Division.

Önal, H. (1993). A modified simplex approach for solving bilevel linear programming problems. *European J. Oper. Res.*, 67:126–135.

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.

Scheel, H. and Scholtes, S. (2000). Mathematical program with complementarity constraints: Stationarity, optimality and sensitivity. *Math. of Oper. Res.*, 25:1–22.

Scholtes, S. (2001). Convergence properties of a regularization scheme for mathematical programs with complementarity constraints. *SIAM J. Optim.*, 11:918–936.

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.

Ye, J. J. (1999). Optimality conditions for optimization problems with complementarity constraints. *SIAM J. Optim.*, 9(2):374–387.

Ye, J. J. (2005). Necessary and sufficient optimality conditions for mathematical programs with equilibrium constraints. *J. Math. Anal. Appl.*, 307:350–369.