Math Meth Oper Res (2018) 87:109–137
https://doi.org/10.1007/s00186-017-0610-4

ORIGINAL ARTICLE

# The double pivot simplex method

Fabio Vitor[1] · Todd Easton[1]

Received: 15 September 2016 / Published online: 23 September 2017

**Abstract** The simplex method, created by George Dantzig, optimally solves a linear program by pivoting. Dantzig's pivots move from a basic feasible solution to a different basic feasible solution by exchanging exactly one basic variable with a nonbasic variable. This paper introduces the double pivot simplex method, which can transition between basic feasible solutions using two variables instead of one. Double pivots are performed by identifying the optimal basis in a two variable linear program using a new method called the slope algorithm. The slope algorithm is fast and allows an iteration of DPSM to have the same theoretical running time as an iteration of the simplex method. Computational experiments demonstrate that DPSM decreases the average number of pivots by approximately 41% on a small set of benchmark instances.

**Keywords** Linear programming · Simplex method · Block pivots · Multiple pivots · Double pivots

**Mathematics Subject Classification** 90C05 · 90C49

✉ Fabio Vitor
fabioftv@ksu.edu

Todd Easton
teaston@ksu.edu

[1] Department of Industrial and Manufacturing Systems Engineering, Kansas State University, 2061 Rathbone Hall, Manhattan, KS 66506, USA

## 1 Introduction

In 1947, George Dantzig created the simplex method, which optimally solves a linear program (Dantzig 1947, 1982). The simplex method is one of the most famous and important developments in science and was recognized by the *Journal of Computing in Science and Engineering* as one of the top 10 algorithms of the twentieth century (Dongarra and Sullivan 2000). This paper improves the simplex method by constructing an efficient algorithm to perform double pivots.

Some of the earliest applications of linear programs were developed by Nobel Laureates (Kantorovich 1939; Koopmans 1949). The simplex method changed linear programs from an interesting model to a practical tool (Dorfman 1984). Currently, linear programs dramatically impact society by finding better solutions to problems in numerous industries including logistics (Spitter et al. 2005; Kunnumkal et al. 2012; García et al. 2013), finance (Chalermkraivuth et al. 2005; Mansini et al. 2007; Nadarajah et al. 2015), manufacturing (Tang et al. 2000; Gomes and Oliveira 2006; Rong and Lahdelma 2008), medicine (Lee et al. 2003; Alterovitz et al. 2006; Romeijn et al. 2006), and governmental policy (Gautier et al. 2000; Bartolini et al. 2007; Zhou and Ang 2008). It suffices to state that millions of linear programs are solved every day, which helps create a more efficient world.

The practical importance of solving linear programs faster has motivated researchers to pursue numerous advancements. The first major advancement occurred when Dantzig and Orchard-Hays (1954) created the revised simplex method. The revised simplex method is an identical algorithm, but requires less computational time because it utilizes matrix operations. The running time of the revised simplex method has a worst-case bound of $O(2^n)$ (Klee and Minty 1972). However, Spielman and Teng (2004) demonstrated that the revised simplex method may run in polynomial time when instances are randomly generated and slightly perturbed.

The next major development occurred when Khachiyan (1979) created a polynomial time ellipsoid algorithm to solve linear programs. Due to the high dependency on the input data, Khachiyan's ellipsoid algorithm proved to be ineffective in practice (Goldfarb and Todd 1989). Next, Karmarkar (1984) proposed the first polynomial time algorithm to solve linear programs that is effective in practice. Numerous researchers have proposed alternate interior point algorithms and one of the computationally fastest algorithms is the infeasible primal-dual (Kojima et al. 1989; Megiddo 1989; Mehrotra 1992; Kojima et al. 1993; Lustig et al. 1994; Gondzio 2012). Currently, there still exists a debate as to whether or not interior point algorithms are computationally faster than the simplex method (Terlaky and Zhang 1993; Bertsimas and Tsitsiklis 1997; Illés and Terlaky 2002; Gondzio 2012).

Numerous other researchers developed computational improvements to the simplex method. Exploiting the sparseness of matrices (Tolla 1986; Suhl and Suhl 1990), handling degeneracy with reduced basis (Elhallaoui et al. 2010; Raymond et al. 2010), and implementing decomposition methods (Ford and Fulkerson 1958; Dantzig and Wolfe 1960; Gilmore and Gomory 1961, 1963; Appelgren 1969) have all improved the computational performance of the simplex method.

From a high level perspective, the simplex method starts with a feasible basis and pivots to another feasible basis. A pivot, which is called a classic pivot in this paper, exchanges exactly one element in the basis with an element that is not in the basis. This paper introduces a double pivot, which can exchange two elements into the basis instead of one. The slope algorithm is a new method that quickly determines the leaving basis elements by finding the optimal basis of a two variable linear program. Double pivots are guaranteed to improve the objective function value by at least as much as classic pivots. Implementing double pivots instead of classic pivots in a simplex framework is called the double pivot simplex method. Performing an iteration of the double pivot simplex method requires the same theoretical effort as an iteration of the simplex method. Computational experiments demonstrate that the double pivot simplex method has on average 41% fewer pivots than the simplex method on a small set of benchmark instances.

The remainder of the paper is organized as follows. Section 2 provides some background information about linear programming. The slope algorithm is presented in Sect. 3, and the double pivot simplex method is presented in Sect. 4. Sections 5 and 6 contain a computational study, conclusions, and future research topics.

## 2 Linear programming basics and background information

The necessary background information to understand this paper is taught in numerous undergraduate and graduate courses in various disciplines worldwide. Additional information is found in Winston (2004), Bazaraa et al. (2009), and Hillier and Lieberman (2015).

A linear program (LP) with $n'$ variables and $r$ constraints takes the form of maximize $z' = c'^T x'$ subject to $A'x' \leq b$ and $x' \geq 0$, where $n'$ and $r$ are positive integers, $c' \in \mathbb{R}^{n'}$, $x' \in \mathbb{R}^{n'}$, $A' \in \mathbb{R}^{r \times n'}$, and $b \in \mathbb{R}^r$. Let $N' = \{1, \ldots, n\}$ be the set of variable indices and $R = \{1, \ldots, r\}$ be the set of constraint indices. The feasible region of an LP is denoted by $S' = \{x' \in \mathbb{R}^{n'}_+ : A'x' \leq b\}$ and the optimal solution of an LP is $(z'^*, x'^*)$, where $x'^* \in S$ and $z'^* = c'^T x'^* \geq c'^T x''$ for all $x'' \in S$.

Given an LP, the simplex method (SM) requires that all constraints be converted into linear equations by adding $r$ slack variables. This new problem is called the standard linear program (SLP) and is defined as maximize $z = c^T x$ subject to $Ax = b$ and $x \geq 0$, where $x \in \mathbb{R}^{n+r}$, $c \in \mathbb{R}^{n+r}$ is $c'$ augmented with $r$ zeros, and $A \in \mathbb{R}^{r \times (n+r)}$ is $A'$ augmented with an $r \times r$ identity matrix. Let $N = \{1, \ldots, n + r\}$ be the variable indices and $S = \{x \in \mathbb{R}^{n+r}_+ : Ax = b\}$ be the feasible region of an SLP.

Initially, SM requires a starting basic feasible solution. Formally, $BV \subseteq N$ is called a basis if $|BV| = |R|$ and $A_{.BV}$ is nonsingular. The set of nonbasic indices is $NBV = N \setminus BV$. The corresponding basic and nonbasic variables are $x_{BV}$ and $x_{NBV}$, respectively. If $A_{.BV}{}^{-1}b \geq 0$, then $BV$ is a feasible basis with $x_{BV} = A_{.BV}{}^{-1}b$ and $x_{NBV} = 0$ being the corresponding basic feasible solution. This paper follows the common notation that a dot "." represents all the columns or rows of a given vector or matrix. Furthermore, a set as a subscript restricts the matrix or vector to only those indices of the set. Therefore, $A_{.BV}$ represents the columns of $A$ restricted to the indices in $BV$, and $x_{BV}$ is the $x$ values of the indices in $BV$. In addition, since SM exchanges an element in $BV$ with an element in $NBV$, order is important and every basis in the remainder of this paper is viewed as an $r$ tuple.

The input to SM is an SLP and a feasible basis $BV \subseteq N$. The algorithm evaluates each nonbasic variable's reduced cost, which is equal to $c_{BV}{}^{T} A_{.BV}{}^{-1} A_{.i} - c_i$ for all $i \in NBV$. If all nonbasic reduced costs are nonnegative, then the corresponding basic feasible solution and $z = c_{BV}{}^{T} x_{BV}$ represents an optimal solution to SLP. Furthermore, $BV$ is said to be an optimal basis. If $BV$ is not optimal, then there exists an entering variable with index $p \in NBV$ such that $c_p < 0$. In order to determine a leaving variable, define $R^+ = \{j \in R : (A_{.BV}{}^{-1} A_{.p})_j > 0\}$. If $R^+ = \emptyset$, then the problem is unbounded. If not, SM performs the ratio test and identifies $j^* \in R^+$ such that $\frac{(A_{.BV}{}^{-1} b)_{j*}}{(A_{.BV}{}^{-1} A_{.p})_{j*}} \leq \frac{(A_{.BV}{}^{-1} b)_j}{(A_{.BV}{}^{-1} A_{.p})_j}$ for all $j \in R^+$. SM replaces the $j^*$th element in $BV$ with $p$. This process is referred to as a pivot and is called a classic pivot in this paper. The algorithm continues pivoting until an optimal basis to SLP is identified or SLP is shown to be unbounded.

In order to generalize SM for other pivoting algorithms, define a simplex framework as the basic steps of SM independent of the pivoting method. A simplex framework starts with a basis $BV$ and identifies $Q \subseteq NBV$. A new basis $\overline{BV} \subseteq BV \cup Q$ is identified and replaces $BV$. The act of selecting $Q$ and replacing $BV$ with $\overline{BV}$ is called a pivot. The simplex framework repeats this process until an optimal basis to SLP is obtained or SLP is shown to be unbounded.

An alternative view of a classic pivot, which led directly to the development of this research, defines the ratio test as optimally solving an SLP with $r + 1$ variables and $r$ constraints. If one could solve an SLP with $r + 2$ variables about as quickly as the ratio test solves the $r + 1$ problem, then one would expect improved pivots resulting in faster computational performance. The next section describes the slope algorithm, which can rapidly solve SLPs with $r + 2$ variables.

## 3 The slope algorithm

One of the simplest LPs has only two variables. These LPs can be easily solved by the graphical method, which is commonly presented in courses teaching linear programming. Many authors have proposed efficient algorithms to solve such simple problems. For instance, Shamos and Hoey (1976) present a $O(rlog(r))$ algorithm that can solve two variable LPs while Megiddo (1983) and Dyer (1984) present linear time algorithms to solve two or three variable LPs. Even though these methods can rapidly find the optimal solution of a two variable LP, such algorithms do not necessarily determine the optimal basis. For instance, if the two variable LP has three or more constraints intersecting the optimal solution, then these methods do not necessarily obtain the optimal basis. If on the other hand, exactly two constraints intersect the optimal solution, then these algorithms can be easily modified to return the optimal basis. Since an iteration of the double pivot simplex method requires the optimal basis from a two variable LP and not the optimal solution, this section presents the slope algorithm (SA), a new method that identifies both the optimal basis and the optimal solution of a two variable LP.

Since SA is implemented within a simplex framework, these two variable LPs satisfy three conditions: the cost coefficients of both variables are positive, both of the variables have nonnegativity constraints, and the right-hand side of every constraint is nonnegative. Formally, let $c_1, c_2 \in \mathbb{R}_+ \setminus \{0\}$, $A \in \mathbb{R}^{r \times 2}$, and $b \in \mathbb{R}^r_+$. Define a simplex two variable linear program (S2LP) as maximize $z = c_1x_1 + c_2x_2$, subject to $a_{j,1}x_1 + a_{j,2}x_2 \leq b_j$ for all $j \in R$, $x_1 \geq 0$, and $x_2 \geq 0$. Define $S^2 = \{x \in \mathbb{R}^2_+ : a_{j,1}x_1 + a_{j,2}x_2 \leq b_j \ \forall\ j \in R\}$ to be the feasible region of an S2LP.

In order to implement SA, the nonnegativity constraints are converted into less than or equal to constraints. Define a slope algorithm two variable linear program (SA2LP) as maximize $z = c_1x_1 + c_2x_2$, subject to $a_{j,1}x_1 + a_{j,2}x_2 \leq b_j$ for all $j \in R' = \{1, \ldots, r+2\}$, where $c_1 > 0$, $c_2 > 0$, $b_j \geq 0$ for all $j \in R'$, $a_{r+1,2} = a_{r+2,1} = 0$, $a_{r+1,1} = a_{r+2,2} = -1$, and $b_{r+1} = b_{r+2} = 0$.

SA evaluates the angle or "slope" of each constraint and compares it to the slope formed by the cost coefficients $c_1$ and $c_2$. Each constraint $j \in R'$ is partitioned into one of nine sets, and each set assigns a value for a slope coefficient, $\alpha_j$, as follows:

$$\alpha_j = \begin{cases} -2M & \text{If } j \in R'_{=<} \text{ where } R'_{=<} = \{j \in R' : a_{j,1} = 0, a_{j,2} < 0\} \\ -M + \frac{a_{j,2}}{a_{j,1}} & \text{If } j \in R'_{><} \text{ where } R'_{><} = \{j \in R' : a_{j,1} > 0, a_{j,2} < 0\} \\ -M & \text{If } j \in R'_{>=} \text{ where } R'_{>=} = \{j \in R' : a_{j,1} > 0, a_{j,2} = 0\} \\ \frac{a_{j,2}}{a_{j,1}} & \text{If } j \in R'_{>>} \text{ where } R'_{>>} = \{j \in R' : a_{j,1} > 0, a_{j,2} > 0\} \\ M & \text{If } j \in R'_{=>} \text{ where } R'_{=>} = \{j \in R' : a_{j,1} = 0, a_{j,2} > 0\} \\ M - \frac{a_{j,1}}{a_{j,2}} & \text{If } j \in R'_{<>} \text{ where } R'_{<>} = \{j \in R' : a_{j,1} < 0, a_{j,2} > 0\} \\ 2M & \text{If } j \in R'_{<=} \text{ where } R'_{<=} = \{j \in R' : a_{j,1} < 0, a_{j,2} = 0\} \\ 3M & \text{If } j \in R'_{==} \text{ where } R'_{==} = \{j \in R' : a_{j,1} = 0, a_{j,2} = 0\} \\ 3M & \text{If } j \in R'_{<<} \text{ where } R'_{<<} = \{j \in R' : a_{j,1} < 0, a_{j,2} < 0\}. \end{cases}$$

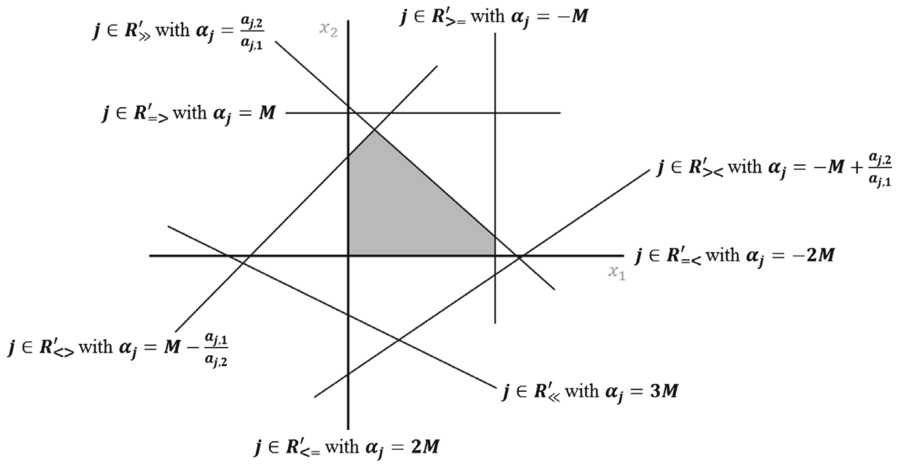
Assign $M > \max\left\{\max_{j \in R'}\left\{\left|\frac{a_{j,1}}{a_{j,2}}\right| : a_{j,2} \neq 0\right\}, \max_{j \in R'}\left\{\left|\frac{a_{j,2}}{a_{j,1}}\right| : a_{j,1} \neq 0\right\}, \frac{c_2}{c_1}\right\}$. Due to the large value of $M$, viewing the constraints in ascending order according to the values of the $\alpha_j$'s creates a counterclockwise orientation of the constraints' slopes starting from the $x_1$ axis. Figure 1 depicts the constraints with their respective $\alpha_j$ values. Observe that only eight constraints are viewable because $R'_{==}$ defines the entire two dimensional space.

The first step in creating SA determines whether or not an SA2LP is unbounded. The following lemma provides a relationship between the coefficients from two constraints, which helps derive conditions of an unbounded SA2LP.

**Lemma 1** *If an SA2LP has $j$ and $k \in R'$ such that $\alpha_j < \alpha_k$ and $\alpha_k \leq -M$, then $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$.*

*Proof* Assume an SA2LP has $j$ and $k \in R'$ such that $\alpha_j < \alpha_k$ and $\alpha_k \leq -M$. If $\alpha_j = -2M$, then $a_{j,1} = 0$, $a_{j,2} < 0$, and $-2M < \alpha_k \leq -M$. Thus, $a_{k,1} > 0$. Consequently, $a_{j,2}a_{k,1} < 0$, $a_{k,2}a_{j,1} = 0$, and $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$.

If $-2M < \alpha_j < -M$, then $a_{j,1} > 0$, $a_{j,2} < 0$, and $\alpha_j < \alpha_k \leq -M$. If $\alpha_k < -M$, then $-M + \frac{a_{j,2}}{a_{j,1}} < -M + \frac{a_{k,2}}{a_{k,1}}$, which results in $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$. If $\alpha_k = -M$,



$j \in R'_{\gg}$ with $\alpha_j = \frac{a_{j,2}}{a_{j,1}}$
$x_2$
$j \in R'_{>=}$ with $\alpha_j = -M$
$j \in R'_{=>}$ with $\alpha_j = M$
$j \in R'_{><}$ with $\alpha_j = -M + \frac{a_{j,2}}{a_{j,1}}$
$j \in R'_{=<}$ with $\alpha_j = -2M$
$x_1$
$j \in R'_{<>}$ with $\alpha_j = M - \frac{a_{j,1}}{a_{j,2}}$
$j \in R'_{\ll}$ with $\alpha_j = 3M$
$j \in R'_{<=}$ with $\alpha_j = 2M$

**Fig. 1** Sample $\alpha_j$ for eight classes of constraints of SA2LP

then $a_{k,1} > 0$ and $a_{k,2} = 0$. Thus, $a_{j,2}a_{k,1} < 0$ and $a_{k,2}a_{j,1} = 0$. Consequently, $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$. □

Since the origin $(0, 0)$ is feasible, $c_1 > 0$, $c_2 > 0$, $x_1 \geq 0$, and $x_2 \geq 0$ for every SA2LP, any feasible ray $d = (d_1, d_2) \in \mathbb{R}^2$ implies that SA2LP is unbounded. Therefore, if there exists $d = (d_1, d_2) \in \mathbb{R}^2_+$ such that $a_{j,1}d_1 + a_{j,2}d_2 \leq 0$ for all $j \in R'$, then SA2LP is unbounded. The following theorem provides necessary and sufficient conditions for an unbounded SA2LP.

**Theorem 1** *An SA2LP is unbounded if and only if the following three conditions hold:*

(i) $R'_{>>} = \emptyset$;
(ii) *If* $R'_{>=} \neq \emptyset$, *then* $R'_{=>} = \emptyset$ *and* $R'_{<>} = \emptyset$;
(iii) *If* $R'_{><} \neq \emptyset$, *then* $R'_{=>} = \emptyset$ *and* $\frac{a_{j,2}}{a_{j,1}} \leq \frac{a_{k,2}}{a_{k,1}}$ *for every* $j \in R'_{><}$ *and every* $k \in R'_{<>}$.

*Proof* Assume an SA2LP is unbounded, then there exists a ray $d = (d_1, d_2)$ such that $a_{j,1}d_1 + a_{j,2}d_2 \leq 0$ for all $j \in R'$ and $c_1d_1 + c_2d_2 > 0$. Since $x_1 \geq 0$, $x_2 \geq 0$, $c_1 > 0$, and $c_2 > 0$, the unbounded ray must satisfy $d_1 \geq 0$, $d_2 \geq 0$, and $d_1 + d_2 > 0$.

Assume there exists a $j \in R'_{>>}$, then $a_{j,1} > 0$ and $a_{j,2} > 0$. Evaluating $d$ on the $j$th constraint results in $a_{j,1}d_1 + a_{j,2}d_2 > 0$, contradicting $d$ being a feasible ray. Thus, $R'_{>>} = \emptyset$ and (i) is satisfied.

Assume there exists a $j \in R'_{>=}$, then $a_{j,1} > 0$ and $a_{j,2} = 0$. Since $d_1 \geq 0$, $d_1 = 0$ or $d$ is not a feasible ray for the $j$th constraint. Thus, $d = (0, d_2)$ where $d_2 > 0$. If any $k \in R'$ has $a_{k,2} > 0$, then $d$ is not a feasible direction for the $k$th constraint. Thus, $R'_{=>} = \emptyset$ and $R'_{<>} = \emptyset$, which satisfies (ii).

Assume there exists a $j \in R'_{><}$, then $a_{j,1} > 0$ and $a_{j,2} < 0$. For contradiction, assume $k \in R'_{=>}$, which implies $a_{k,1} = 0$ and $a_{k,2} > 0$. In order for $d$ to be an improving direction that is feasible on the $k$th constraint, $d_2 = 0$ and $d_1 > 0$. Thus,

$a_{j,1}d_1 + a_{j,2}d_2 > 0$, contradicting $d$ being a feasible direction for the $j$th constraint. Consequently, $R'_{=>} = \emptyset$ and the first condition of (iii) is satisfied.

Assume there exists $j \in R'_{><}$ and $k \in R'_{<>}$, which implies $a_{j,1} > 0$, $a_{j,2} < 0$, $a_{k,1} < 0$, and $a_{k,2} > 0$. Since $d$ is an improving direction and feasible on the $j$th and $k$th constraints, $d_1 > 0$, $d_2 > 0$, $a_{j,1}d_1 + a_{j,2}d_2 \leq 0$, and $a_{k,1}d_1 + a_{k,2}d_2 \leq 0$. Therefore, $d_1 \leq -\frac{a_{j,2}d_2}{a_{j,1}}$. Substituting into the second inequality results in $a_{k,1}\big(-\frac{a_{j,2}d_2}{a_{j,1}}\big) + a_{k,2}d_2 \leq 0$, which implies $d_2\Big(-\frac{a_{k,1}a_{j,2}}{a_{j,1}} + a_{k,2}\Big) \leq 0$. Since $d_2 > 0$, $-\frac{a_{k,1}a_{j,2}}{a_{j,1}} + a_{k,2} \leq 0$, implying that $\frac{a_{j,2}}{a_{j,1}} \leq \frac{a_{k,2}}{a_{k,1}}$. Therefore, the second condition of (iii) is satisfied.

Conversely, assume an SA2LP satisfies conditions (i), (ii), and (iii). Let $j^* \in R'_{=<} \cup R'_{><} \cup R'_{>=}$ such that $\alpha_{j^*} \geq \alpha_j$ for all $j \in R'_{=<} \cup R'_{><} \cup R'_{>=}$. The claim is that the $j^*$th constraint defines a ray of unboundedness, which is $d = (-a_{j^*,2}, a_{j^*,1})$. Trivially, $c_1(-a_{j^*,2}) + c_2(a_{j^*,1}) > 0$ for any $j^* \in R'_{=<} \cup R'_{><} \cup R'_{>=}$. Since the origin is feasible, it suffices to show that $d$ is a feasible direction for each $k \in R'$.

Let $k \in R'$ such that $\alpha_k < \alpha_{j^*}$. The conditions of Lemma 1 are satisfied and $a_{k,2}a_{j^*,1} < a_{k,1}a_{j^*,2}$, which implies $a_{k,2}a_{j^*,1} - a_{k,1}a_{j^*,2} < 0$. If $k \in R'$ such that $\alpha_k = \alpha_{j^*}$, then the $j^*$th and $k^*$th constraints are parallel and $a_{k,2}a_{j^*,1} - a_{k,1}a_{j^*,2} = 0$. Consequently, $d$ is a feasible direction for every such constraint and the remainder of the cases need only consider $\alpha_k > \alpha_{j^*}$.

If $j^* \in R'_{=<}$, then $R'_{><} = \emptyset$ and $R'_{>=} = \emptyset$ because $\alpha_{j^*} \geq \alpha_j$ for all $j \in R'_{=<} \cup R'_{><} \cup R'_{>=}$. Since $R'_{>>} = \emptyset$, $a_{k,1} \leq 0$ for all $k \in R'$. Thus, $a_{k,1}(-a_{j^*,2}) + a_{k,2}(0) \leq 0$ for all $k \in R'$ and $d$ is a feasible improving ray.

If $j^* \in R'_{><}$, $a_{j^*,2} < 0$, $a_{j^*,1} > 0$, $R'_{>>} = \emptyset$, $R'_{=>} = \emptyset$, and $R'_{>=} = \emptyset$ due to (i), the first condition of (iii), and $\alpha_{j^*}$ being the maximum $\alpha_j$ for all $j \in R'_{=<} \cup R'_{><} \cup R'_{>=}$. For any $k \in R'_{<=} \cup R'_{==} \cup R'_{<<}$, $a_{k,1}(-a_{j^*,2}) \leq 0$ and $a_{k,2}(a_{j^*,1}) \leq 0$, so $a_{k,1}(-a_{j^*,2}) + a_{k,2}(a_{j^*,1}) \leq 0$. If $k \in R'_{<>}$, $a_{k,1} < 0$. The second condition of (iii), ($\frac{a_{j^*,2}}{a_{j^*,1}} \leq \frac{a_{k,2}}{a_{k,1}}$ for all $k \in R'_{<>}$), implies $a_{j^*,2}a_{k,1} \geq a_{k,2}a_{j^*,1}$. Therefore, $a_{k,1}(-a_{j^*,2}) + a_{k,2}(a_{j^*,1}) \leq 0$. Consequently, $d$ is a feasible direction for each $k \in R'$ such that $\alpha_k > \alpha_{j^*}$. Thus, $d$ is a ray of unboundedness.

If $j^* \in R'_{>=}$, then $R'_{>>} = \emptyset$, $R'_{=>} = \emptyset$, and $R'_{<>} = \emptyset$ according to (i) and (ii). Therefore, $a_{k,2} \leq 0$ and $a_{k,1}(0) + a_{k,2}(a_{j^*,1}) \leq 0$ for all $k \in R'$ such that $\alpha_k > \alpha_{j^*}$. Consequently, $d$ is a feasible improving ray. Since all cases have an improving ray, SA2LP is unbounded. □

The three graphs in Fig. 2 illustrate the conditions of Theorem 1. The $j^*$ constraint is labeled in each figure and is represented by the solid line. This constraint identifies a ray of unboundedness as shown in the theorem. The dashed lines demonstrate constraints that cannot exist for SA2LP to be unbounded.

Since the origin is feasible, an optimal solution to SA2LP exists whenever SA2LP is bounded. The following lemma and two corollaries provide other useful relationships between the coefficients of two constraints and also the objective coefficients $c_1$ and $c_2$. From any SA2LP, define SA2LP$_{j,k}$ to be an SA2LP with only four constraints. The constraints are the two nonnegativity constraints and the $j$th and $k$th constraints from SA2LP.

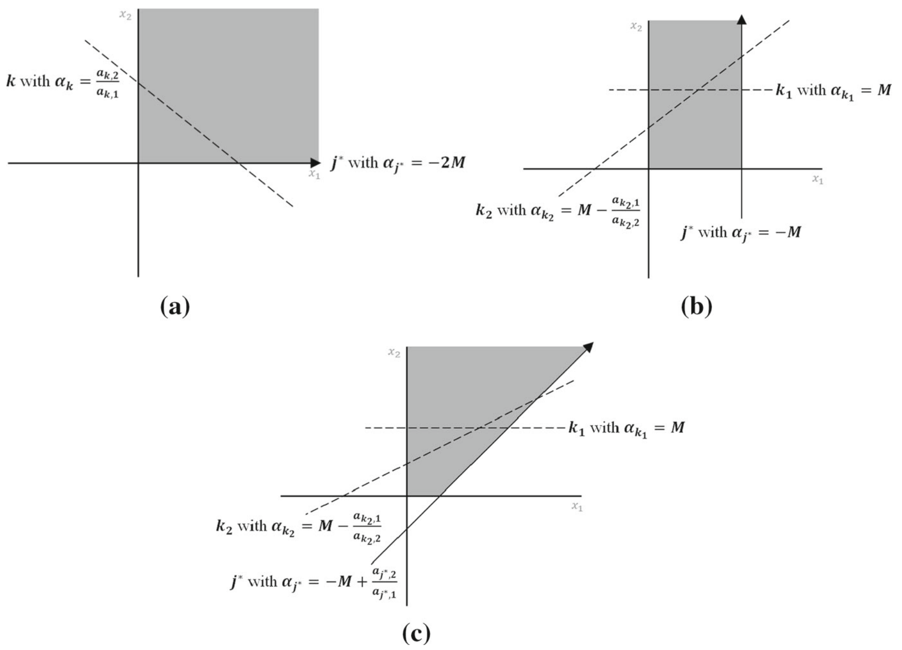

$x_2$
$x_1$
$k$ with $\alpha_k = \frac{a_{k,2}}{a_{k,1}}$
$j^*$ with $\alpha_{j^*} = -2M$
(a)
$k_1$ with $\alpha_{k_1} = M$
$k_2$ with $\alpha_{k_2} = M - \frac{a_{k_2,1}}{a_{k_2,2}}$
$j^*$ with $\alpha_{j^*} = -M$
(b)
$j^*$ with $\alpha_{j^*} = -M + \frac{a_{j^*,2}}{a_{j^*,1}}$
(c)

**Fig. 2** Graphical representation of Theorem 1—**a** depicts condition (i), **b** represents condition (ii), and **c** corresponds to condition (iii)

**Lemma 2** *If an SA2LP has $j$ and $k \in R'$ such that SA2LP$_{j,k}$ is bounded, $\alpha_j < M$, $-M < \alpha_k \leq 2M$, and $\alpha_j < \alpha_k$, then $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$.*

*Proof* Assume an SA2LP has $j$ and $k \in R'$ such that SA2LP$_{j,k}$ is bounded, $\alpha_j < M$, $-M < \alpha_k \leq 2M$, and $\alpha_j < \alpha_k$. From Theorem 1, the potential combinations for the $j$ and $k$ constraints are limited. The proof shows that $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$ for all possible values of $\alpha_j$.

If $\alpha_j = -2M$, then $a_{j,1} = 0$ and $a_{j,2} < 0$. Since SA2LP$_{j,k}$ is bounded and from the conditions of Theorem 1, $\alpha_k < M$. So $a_{k,1} > 0$ and $a_{k,2} > 0$. Consequently, $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$.

If $-2M < \alpha_j < -M$, then $a_{j,1} > 0$, $a_{j,2} < 0$, and $\alpha_k < 2M$ or SA2LP$_{j,k}$ is unbounded from Theorem 1. If $\alpha_k \leq M$, then $a_{k,1} > 0$ and $a_{k,2} \geq 0$. Thus, $a_{j,2}a_{k,1} < 0$ and $a_{j,1}a_{k,2} \geq 0$, which implies $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$. If $\alpha_k > M$ and because SA2LP$_{j,k}$ is bounded, $\frac{a_{j,2}}{a_{j,1}} > \frac{a_{k,2}}{a_{k,1}}$ by Theorem 1. Hence, $a_{j,2}a_{k,1} < a_{k,2}a_{j,1}$.

If $\alpha_j = -M$, then $a_{j,1} > 0$, $a_{j,2} = 0$, and $\alpha_k < 2M$ or SA2LP$_{j,k}$ is unbounded. Thus, $a_{k,2} > 0$, $a_{j,2}a_{k,1} = 0$, and $a_{j,1}a_{k,2} > 0$. Therefore, $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$.

If $-M < \alpha_j < M$, then $a_{j,1} > 0$ and $a_{j,2} > 0$. If $\alpha_k < M$, then $\frac{a_{j,2}}{a_{j,1}} < \frac{a_{k,2}}{a_{k,1}}$ and $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$, because $\alpha_j < \alpha_k$. If $M \leq \alpha_k < 2M$, then $a_{k,1} \leq 0$ and $a_{k,2} > 0$. Consequently, $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$. If $\alpha_k = 2M$, then $a_{k,1} < 0$ and $a_{k,2} = 0$, and so $a_{j,2}a_{k,1} < a_{j,1}a_{k,2} = 0$. □

Observe that the right-hand side $b_j$ and $b_k$ are not contained in the proofs or statements of Lemmas 1 and 2. These lemmas are based solely on the slopes, represented by $\alpha_j$ and $\alpha_k$. Thus, these relationships remain between a constraint and the cost coefficients $c_1$ and $c_2$. Since $c_1 > 0$ and $c_2 > 0$, the $\alpha$ value of the objective function is equivalent to $\frac{c_2}{c_1}$. Since $-M < \frac{c_2}{c_1} < M$, the following two corollaries are direct applications of Lemma 2.

**Corollary 1** *If an SA2LP has a $j \in R'$ such that $\alpha_j < \frac{c_2}{c_1}$, then $a_{j,2}c_1 < a_{j,1}c_2$.* $\square$

**Corollary 2** *If an SA2LP has a $k \in R'$ such that $\frac{c_2}{c_1} \leq \alpha_k \leq 2M$, then $c_2 a_{k,1} \leq c_1 a_{k,2}$.*

*Proof* Assume an SA2LP has a $k \in R'$ such that $\frac{c_2}{c_1} < \alpha_k$. From Lemma 2, $c_2 a_{k,1} < c_1 a_{k,2}$. If $\frac{c_2}{c_1} = \alpha_k$, then $a_{k,1} > 0$ and $a_{k,2} > 0$. Thus, $\frac{c_2}{c_1} = \frac{a_{k,2}}{a_{k,1}}$ and $c_2 a_{k,1} = c_1 a_{k,2}$, which implies $c_2 a_{k,1} \leq c_1 a_{k,2}$. $\square$

There are typically an infinite number of points in $\mathbb{R}^2$ that satisfy both the $j$th and $k$th constraints. This paper defines the intersection point of the $j$th and $k$th constraints as the unique extreme point of the feasible region defined by only these two constraints. If no such extreme point exists, then the $j$th and $k$th constraints are parallel, $a_{j,1}a_{k,2} - a_{j,2}a_{k,1} = 0$, and the constraints are said to be nonintersecting. Consequently, any intersecting constraints satisfy $a_{j,1}a_{k,2} - a_{j,2}a_{k,1} \neq 0$, and the intersection point is given by $\overline{x} = (\overline{x_1}, \overline{x_2})$, where $\overline{x_1} = \frac{b_j a_{k,2} - b_k a_{j,2}}{a_{j,1}a_{k,2} - a_{j,2}a_{k,1}}$ and $\overline{x_2} = \frac{a_{j,1}b_k - b_j a_{k,1}}{a_{j,1}a_{k,2} - a_{j,2}a_{k,1}}$.

Theorem 2 identifies nonsupportive constraints in $S^2$ by evaluating $\overline{x}$ on a particular constraint. A constraint is said to support a polyhedron if there exists a point in the polyhedron that meets the constraint at equality.

**Theorem 2** *If a bounded SA2LP has $i$, $j$, and $k \in R'$ such that $\alpha_i < \alpha_j < \alpha_k \leq 2M$, $\alpha_j < M$, and the intersection point of the $j$th and $k$th constraints violates the $i$th constraint, then the $j$th constraint does not support $S^2$.*

*Proof* Assume an SA2LP is bounded and there exists constraints $i$, $j$, and $k \in R'$ such that $\alpha_i < \alpha_j < \alpha_k \leq 2M$, $\alpha_j < M$, and the intersection point of the $j$th and $k$th constraints, $\overline{x}$, violates the $i$th constraint. The points in $\mathbb{R}^2$ that meet the $j$th constraint at equality can be expressed as $\overline{x} + \rho(a_{j,2}, -a_{j,1})$ and $\overline{x} + \lambda(-a_{j,2}, a_{j,1})$ where $\rho \geq 0$ and $\lambda > 0$.

Evaluating $\overline{x} + \rho(a_{j,2}, -a_{j,1})$ for all $\rho \geq 0$ on the $i$th constraint results in $a_{i,1}\overline{x_1} + a_{i,2}\overline{x_2} + \rho(a_{i,1}a_{j,2} - a_{i,2}a_{j,1})$. Since $\overline{x}$ violates the $i$th constraint, $a_{i,1}\overline{x_1} + a_{i,2}\overline{x_2} > b_i$. The $i$th and $j$th constraints satisfy either the conditions of Lemmas 1 or 2 and $a_{i,2}a_{j,1} < a_{i,1}a_{j,2}$. Therefore, $a_{i,1}\overline{x_1} + a_{i,2}\overline{x_2} + \rho(a_{i,1}a_{j,2} - a_{i,2}a_{j,1}) > b_i$ for all $\rho \geq 0$, and none of these points is in $S^2$.

Evaluating $\overline{x} + \lambda(-a_{j,2}, a_{j,1})$ for all $\lambda > 0$ on the $k$th constraint results in $a_{k,1}\overline{x_1} + a_{k,2}\overline{x_2} + \lambda(-a_{k,1}a_{j,2} + a_{k,2}a_{j,1})$. Since $\overline{x}$ satisfies the $k$th constraint, $a_{k,1}\overline{x_1} + a_{k,2}\overline{x_2} = b_k$. The $j$th and $k$th constraints satisfy either the conditions of Lemmas 1 or 2 and $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$. Thus, $a_{k,1}\overline{x_1} + a_{k,2}\overline{x_2} + \lambda(-a_{k,1}a_{j,2} + a_{k,2}a_{j,1}) > b_k$ for all $\lambda > 0$, and none of these points is in $S^2$. Consequently, the $j$th constraint does not support $S^2$. $\square$

Corollary 3 trivially extends this result to constraints with larger slope values. The proof is obtained by simply swapping $x_1$ and $x_2$ and applying Theorem 2.

**Corollary 3** *If a bounded SA2LP has $j$, $k$, and $l \in R'$ such that $\alpha_j < \alpha_k < \alpha_l \leq 2M$, $-M < \alpha_k$, and the intersection point of the $j$th and $k$th constraints violates the $l$th constraint, then the $k$th constraint does not support $S^2$.* □

The next two corollaries identify intersection points of constraints and apply Theorem 2 and Corollary 3 to determine nonsupporting constraints. Combining these results enables the creation of a linear time procedure to identify a feasible extreme point of $S^2$, assuming $\alpha$ is sorted.

**Corollary 4** *If a bounded SA2LP has $h, i, j$, and $k \in R'$ such that $\alpha_h < \alpha_i < \alpha_j < \alpha_k \leq 2M$, $\alpha_j < M$, and the intersection point of the $j$th and $k$th constraints satisfies the $i$th constraint but violates the $h$th constraint, then the $i$th and $j$th constraints do not support $S^2$.*

*Proof* Assume a bounded SA2LP has constraints $h, i, j$, and $k \in R'$ such that $\alpha_h < \alpha_i < \alpha_j < \alpha_k \leq 2M$, $\alpha_j < M$, the intersection point of the $j$th and $k$th constraints, $\overline{x}$, satisfies the $i$th constraint, and violates the $h$th constraint. Let the intersection point of the $i$th and $k$th constraints be $\overline{\overline{x}}$. Since $\overline{x}$ satisfies the $i$th constraint, $\overline{\overline{x}} = \overline{x} + \lambda(a_{k,2}, -a_{k,1})$ for some $\lambda \geq 0$. Evaluating $\overline{\overline{x}}$ on the $h$th constraint results in $a_{h,1}\overline{x_1} + a_{h,2}\overline{x_2} + \lambda(a_{h,1}a_{k,2} - a_{h,2}a_{k,1})$. Since $\overline{x}$ violates the $h$th constraint, $a_{h,1}\overline{x_1} + a_{h,2}\overline{x_2} > b_h$. The $h$th and $k$th constraints satisfy either the conditions of Lemma 1 or Lemma 2 and $a_{h,2}a_{k,1} < a_{h,1}a_{k,2}$. Thus, $a_{h,1}\overline{x_1} + a_{h,2}\overline{x_2} + \lambda(a_{h,1}a_{k,2} - a_{h,2}a_{k,1}) > b_h$. Consequently, $\overline{\overline{x}}$ violates the $h$th constraint and the conditions of Theorem 2 are satisfied, which implies that the $i$th and $j$th constraints do not support $S^2$. □

**Corollary 5** *If a bounded SA2LP has $j, k, l$, and $m \in R'$ such that $\alpha_j < \alpha_k < \alpha_l < \alpha_m \leq 2M$, $-M < \alpha_k$, and the intersection point of the $j$th and $k$th constraints satisfies the $l$th constraint, but violates the $m$th constraint, then the $l$th and $k$th constraints do not support $S^2$.* □

SA identifies an optimal solution to SA2LP by finding constraints $j$ and $k \in R'$ such that $\alpha_j < \frac{c_2}{c_1} \leq \alpha_k$, SA2LP$_{j,k}$ is bounded, the intersection point of the $j$th and $k$th constraints is feasible, and $\alpha_k - \alpha_j$ is minimized. The following theorem formalizes that the first three conditions are sufficient to identify an optimal solution to SA2LP.

**Theorem 3** *An optimal solution to SA2LP occurs at the intersection point of the $j$th and $k$th constraints if the following three conditions hold:*

(i) $\alpha_j < \frac{c_2}{c_1} \leq \alpha_k \leq 2M$;
(ii) *SA2LP$_{j,k}$ is bounded;*
(iii) *The intersection point of the $j$th and $k$th constraints is feasible.*

*Proof* Assume SA2LP has $j$ and $k \in R'$ such that SA2LP$_{j,k}$ is bounded, $\alpha_j < \frac{c_2}{c_1} \leq \alpha_k \leq 2M$, and the intersection point of the $j$th and $k$th constraints, $\overline{x}$, is feasible. The proof shows that every direction from $\overline{x}$ is either infeasible or nonimproving. So, partition all possible directions in $\mathbb{R}^2 \setminus \{(0, 0)\}$ into the following four sets:

- $D^1 = \{d \in \mathbb{R}^2 \setminus \{(0, 0)\} : d = \beta(a_{j,2}, -a_{j,1}) + (1-\beta)(a_{k,2}, -a_{k,1})\ \forall \beta \in [0, 1)\}$;
- $D^2 = \{d \in \mathbb{R}^2 \setminus \{(0, 0)\} : d = \beta(-a_{j,2}, a_{j,1}) + (1-\beta)(a_{k,2}, -a_{k,1})\ \forall \beta \in (0, 1)\}$;

- $D^3 = \{d \in \mathbb{R}^2 \setminus \{(0,0)\} : d = \beta(-a_{j,2}, a_{j,1}) + (1-\beta)(-a_{k,2}, a_{k,1})\ \forall \beta \in (0,1]\}$;
- $D^4 = \{d \in \mathbb{R}^2 \setminus \{(0,0)\} : d = \beta(a_{j,2}, -a_{j,1}) + (1-\beta)(-a_{k,2}, a_{k,1})\ \forall \beta \in [0,1]\}$.

Since $-M < \frac{c_2}{c_1} \leq \alpha_k$, the $j$th and $k$th constraints satisfy Lemma 2 and $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$. Evaluating any $d \in D^1$ on the $j$th constraint results in $\beta(a_{j,1}a_{j,2}) + (1-\beta)(a_{j,1}a_{k,2}) - \beta(a_{j,2}a_{j,1}) - (1-\beta)(a_{j,2}a_{k,1}) = (1-\beta)(a_{j,1}a_{k,2} - a_{j,2}a_{k,1}) > 0$ for all $\beta \in [0,1)$. Thus, $\overline{x} + \lambda d$ violates the $j$th constraint for every $\lambda > 0$, and $D^1$ is a set of infeasible directions from $\overline{x}$.

Evaluating any $d \in D^2$ on the $j$th constraint results in $-\beta(a_{j,1}a_{j,2}) + (1-\beta)(a_{j,1}a_{k,2}) + \beta(a_{j,2}a_{j,1}) - (1-\beta)(a_{j,2}a_{k,1}) = (1-\beta)(a_{j,1}a_{k,2} - a_{j,2}a_{k,1}) > 0$ for all $\beta \in (0,1)$. Thus, $\overline{x} + \lambda d$ violates the $j$th constraint for every $\lambda > 0$, and $D^2$ is a set of infeasible directions from $\overline{x}$.

Evaluating any $d \in D^3$ on the $k$th constraint results in $-\beta(a_{k,1}a_{j,2}) - (1-\beta)(a_{k,1}a_{k,2}) + \beta(a_{k,2}a_{j,1}) + (1-\beta)(a_{k,2}a_{k,1}) = \beta(-a_{k,1}a_{j,2} + a_{k,2}a_{j,1}) > 0$ for all $\beta \in (0,1]$. Thus, $\overline{x} + \lambda d$ violates the $k$th constraint for every $\lambda > 0$, and $D^3$ is a set of infeasible directions from $\overline{x}$.

Evaluating any $d \in D^4$ on the objective function results in $\beta(c_1 a_{j,2} - c_2 a_{j,1}) + (1-\beta)(-c_1 a_{k,2} + c_2 a_{k,1})$. Since $\alpha_j < \frac{c_2}{c_1}$, the conditions of Corollary 1 are satisfied and $a_{j,2}c_1 < a_{j,1}c_2$. Thus, $\beta(c_1 a_{j,2} - c_2 a_{j,1}) \leq 0$ for all $\beta \in [0,1]$. Since $\frac{c_2}{c_1} \leq \alpha_k$, $c_2 a_{k,1} \leq c_1 a_{k,2}$ by Corollary 2 and $(1-\beta)(-c_1 a_{k,2} + c_2 a_{k,1}) \leq 0$ for all $\beta \in [0,1]$. Consequently, every $d \in D^4$ is a nonimproving direction. Since every direction from $\overline{x}$ is infeasible or nonimproving and SA2LP is a linear convex problem, $\overline{x}$ is an optimal solution. □

The above results enable the creation of SA (Algorithm 1), which optimally solves S2LPs. The input to SA is an S2LP, and SA returns that either S2LP is unbounded or the optimal solution $z^*$, $x^*$ along with $j^* \in R'$, $k^* \in R'$, $\alpha_{j^*}$, and $\alpha_{k^*}$. The $j^*$ and $k^*$ represent the intersecting constraints that provide the optimal solution to S2LP. Even though $j^*$, $k^*$, $\alpha_{j^*}$, and $\alpha_{k^*}$ are not part of S2LP's solution, this information is necessary to identify the optimal basis.

SA correctly solves any SA2LP. The check for unboundedness follows the conditions of Theorem 1 when the constraints are viewed from their $\alpha$ values. If SA2LP is bounded, then the algorithm returns an $x^*$ at the intersection point of two constraints, $j^*$ and $k^*$. Clearly, such a $j^*$ and $k^*$ exist due to the nonnegativity constraints. One constraint has an $\alpha$ value less than $\frac{c_2}{c_1}$ and the other constraint has an $\alpha$ value greater than or equal to $\frac{c_2}{c_1}$. The point is validated against all constraints according to Theorem 2 and Corollaries 3, 4, and 5. From Theorem 3, $x^*$ is an optimal solution to SA2LP.

To determine the running time of SA, observe that SA first calculates every element of the array $\alpha$ in $O(r)$ and sorts this array (lines 3–4). There are numerous sorting algorithms and let $S(r)$ be the effort required by the selected algorithm to sort $r$ elements. The main step of SA (lines 5–28) first determines two intersecting constraints in $O(r)$. The check for unboundedness is performed in $O(1)$. If SA2LP is bounded, then SA calculates $\overline{x}$ in $O(1)$. Each iteration of the while loop either validates that the current $\overline{x}$ is feasible on up to two constraints in $O(1)$ or that $\overline{x}$ violates a constraint. If a violation occurs, a new $\overline{x}$ is calculated. From Corollaries 4 and 5, if $\overline{x}$ violates

**Algorithm 1** - The slope algorithm (SA)

```
1: begin
2:    From S2LP, create the corresponding SA2LP;
3:    Calculate $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_{r+2})$;
4:    Let $P = (\rho_1, \dots, \rho_{r+2})$ be the array of constraint indices such that $\alpha_{\rho_j} \leq \alpha_{\rho_{j+1}} \; \forall \, j \in \{1, \dots, r+1\}$;
5:    Find $j' \in R' = \{1, \dots, r+2\}$ such that $\alpha_{\rho_{j'}} < \frac{c_2}{c_1} \leq \alpha_{\rho_{j'+1}}$;
6:    $k' \leftarrow j' + 1$;
7:    if $\Big(\alpha_{\rho_{j'}} = -2M$ and $\alpha_{\rho_{k'}} \geq M\Big)$ or $\Big(-2M < \alpha_{\rho_{j'}} < -M$ and $\alpha_{\rho_{k'}} = 2M\Big)$ or
   $\Big(\alpha_{\rho_{j'}} = -M$ and $\alpha_{\rho_{k'}} = 2M\Big)$ or
   $\Big(-2M < \alpha_{\rho_{j'}} < -M$ and $M < \alpha_{\rho_{k'}} < 2M$ and $\frac{a_{\rho_{j'},2}}{a_{\rho_{j'},1}} \leq \frac{a_{\rho_{k'},2}}{a_{\rho_{k'},1}}\Big)$ then
8:       return S2LP is unbounded;
9:    else
10:      $j \leftarrow j'$;
11:      $k \leftarrow k'$;
12:      Calculate $\overline{x} = (\overline{x_1}, \overline{x_2})$ from constraints $\rho_{j'}$ and $\rho_{k'}$;
13:      while $j > 1$ or $k < r + 2$ do
14:         if $j > 1$ then $j \leftarrow j - 1$;
15:         if $a_{\rho_j,1}\overline{x_1} + a_{\rho_j,2}\overline{x_2} > b_{\rho_j}$ then
16:            $j' \leftarrow j$;
17:            $k \leftarrow k'$;
18:            Calculate $\overline{x} = (\overline{x_1}, \overline{x_2})$ from $\rho_{j'}$ and $\rho_{k'}$;
19:         if $k < r + 2$ then $k \leftarrow k + 1$;
20:         if $a_{\rho_k,1}\overline{x_1} + a_{\rho_k,2}\overline{x_2} > b_{\rho_k}$ then
21:            $k' \leftarrow k$;
22:            $j \leftarrow j'$;
23:            Calculate $\overline{x} = (\overline{x_1}, \overline{x_2})$ from $\rho_{j'}$ and $\rho_{k'}$;
24:      $z^* \leftarrow c_1\overline{x_1} + c_2\overline{x_2}$;
25:      $x^* \leftarrow \overline{x}$;
26:      $j^* \leftarrow \rho_{j'}$;
27:      $k^* \leftarrow \rho_{k'}$;
28:      return $z^*$, $x^*$, $j^*$, $k^*$, $\alpha_{j^*}$, and $\alpha_{k^*}$;
29: end
```

a constraint, then every constraint between $j'$ and $j$ or $k'$ and $k$ is nonsupportive in SA2LP. Consequently, the while loop is repeated at most $O(r)$ times. Thus, the entire main step requires $O(r)$ effort. Furthermore, SA requires $O(1)$ to report a solution to SA2LP or that SA2LP is unbounded. Consequently, SA requires $O(S(r))$ effort and the most time-consuming step is sorting the array $\alpha$. From merge sort, $S(r) = r\log(r)$ and SA runs in $O(r\log(r))$ time. Example 1 demonstrates SA.

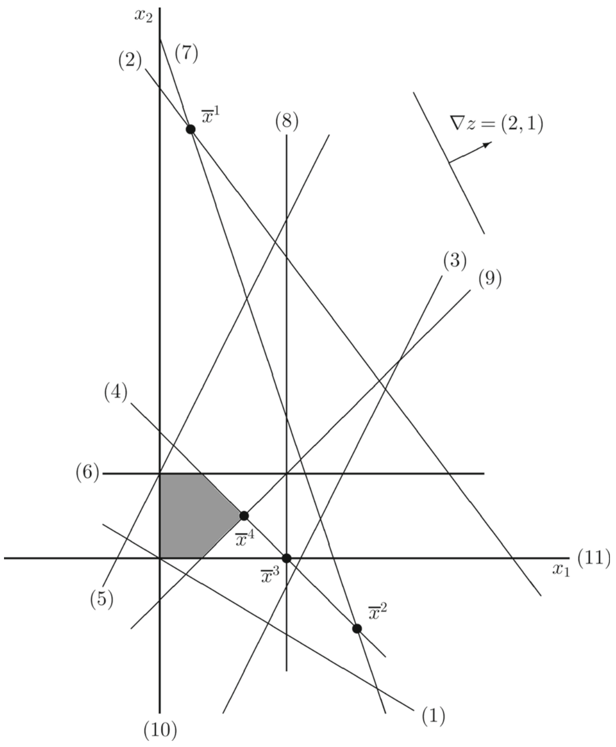


**Fig. 3** Graphical representation of Example 1

*Example 1* Consider the following S2LP.

$$\begin{array}{llr}
\text{Maximize} & z = 2x_1 + x_2 & \\
\text{Subject to} & -3x_1 - 5x_2 \le 0 & (1) \\
& 4x_1 + 3x_2 \le 100 & (2) \\
& 2x_1 - x_2 \le 20 & (3) \\
& x_1 + x_2 \le 9 & (4) \\
& -2x_1 + x_2 \le 6 & (5) \\
& x_2 \le 6 & (6) \\
& 3x_1 + x_2 \le 37 & (7) \\
& x_1 \le 9 & (8) \\
& x_1 - x_2 \le 3 & (9) \\
& x_1, x_2 \ge 0 &
\end{array}$$

The first step of SA converts S2LP into SA2LP by changing the nonnegativity conditions into constraints $-x_1 \le 0$ (10), $-x_2 \le 0$ (11), and assigning $R' = \{1, 2, \dots, 11\}$. Figure 3 presents a graphical representation of SA2LP in $\mathbb{R}^2$. SA calculates $\alpha = (3M, \frac{3}{4}, -M - \frac{1}{2}, 1, M + 2, M, \frac{1}{3}, -M, -M - 1, 2M, -2M)$

and then sorts the indices of the constraints according to these values, resulting in $P = (11, 9, 3, 8, 7, 2, 4, 6, 5, 10, 1)$. SA identifies $j' = 5$ and $k' = 6$ because $\alpha_{\rho_{j'}} = \frac{1}{3} < \frac{c_2}{c_1} = \frac{1}{2} \leq \alpha_{\rho_{j'+1}} = \frac{3}{4}$. Since $-M < \alpha_{\rho_{j'}}$, none of the conditions for an unbounded SA2LP are satisfied. The algorithm continues by setting $j = 5$, $k = 6$, and calculating the intersection point of the $\rho_5$ and $\rho_6$ constraints, (7) and (2). This intersection point is $\overline{x} = \left(\frac{11}{5}, \frac{152}{5}\right)$, represented by $\overline{x}^1$ in Fig. 3.

SA assigns $j = 4$ and the feasibility of $\overline{x}$ is validated on the $\rho_4$ constraint, (8), because $\frac{11}{5} < 9$. Next, $k = 7$ and the point is tested on the $\rho_7$ constraint, (4). This point is infeasible because $\frac{11}{5} + \frac{152}{5} > 9$. From Corollary 3, (2) does not support $S^2$ and SA assigns $k' = 7$ and $j$ returns to 5. From SA, $\overline{x}$ becomes $(14, -5)$, which is the intersection point of (7) and (4) and is represented by $\overline{x}^2$ in Fig. 3. SA updates $j = 4$ and $\overline{x}$ is evaluated on (8), which indicates an infeasibility since $14 > 9$. Thus, $j' = 4$ and $k = 7$. The updated $\overline{x}$ occurs at $(9, 0)$, the intersection of (8) and (4), and is represented by $\overline{x}^3$ in Fig. 3. SA assigns $k = 8$ and $\overline{x}$ does not violate (6), (3), and (5), but it does violate (9). From Theorem 2 and Corollary 4, both (8) and (3) do not support $S^2$. The algorithm assigns $j' = 2$ and $k = 7$. The new $\overline{x} = (6, 3)$ is the intersection point of (9) and (4), and is identified by $\overline{x}^4$ in Fig. 3. SA follows with $k = 8$ and this point does not violate (6), (11), (5), (10), and (1). Thus, $\overline{x} = (6, 3)$ satisfies all the constraints and by Theorem 3, this point is the optimal solution to S2LP. SA reports $z^* = 15$, $x^* = (6, 3)$, $j^* = 9$, $k^* = 4$, $\alpha_9 = -M - 1$, and $\alpha_4 = 1$.

Similar to the dual simplex method, SA starts super optimal and moves toward feasibility while still maintaining the optimality condition. When an $\overline{x}$ is infeasible, one constraint is replaced with another constraint, which maintains a basis structure. From the corresponding basis of the starting constraints, one could create dual simplex pivots to obtain SA's sequence of super optimal solutions. However, the order of pivots would be unique and a standard implementation of the dual simplex method is unlikely to result in the identical sequence of super optimal solutions. Furthermore, the dual simplex method does not necessarily have a starting basis and thus it is unlikely to start with a basis that corresponds to the first $\overline{x}$.

In summary, SA is a new method to solve simple two variable LPs with fairly stringent assumptions. The authors believe that a straightforward exercise can modify SA to solve any two variable LP. For this paper, the true benefit of SA is realized as a pivoting technique incorporated in a simplex framework. The next section discusses the concept of a multiple pivot simplex method and presents the double pivot simplex method.

## 4 Multiple pivots and the double pivot simplex method

A common technique to decrease the solution time of LPs is to solve the problem on a subset of variables. The optimal basis from this subproblem is used to identify a new subset of variables, and the process repeats until the optimal basis to a subproblem identifies the optimal basis of the original instance. In this paper, this general methodology is called a multiple pivot simplex method. Block pivots (Howard 1960; Padberg 1999; Ye 2011), decomposition methods (Dantzig and Wolfe 1960), and column generation (Ford and Fulkerson 1958; Gilmore and Gomory 1961, 1963; Appelgren 1969) can all be classified as multiple pivot simplex methods.

Formally, a multiple pivot simplex method starts with an SLP and a feasible basis $BV \subseteq N$. The method identifies $Q \subseteq NBV$ such that $Q \neq \emptyset$ and $c_{BV}{}^T A_{.BV}{}^{-1} A_{.q} - c_q < 0$ for some $q \in Q$. Define a multiple pivot linear program (MPLP) as maximize $z = -(c_{BV}{}^T A_{.BV}{}^{-1} A_{.Q} - c_Q) x_Q$ subject to $(A_{.BV}{}^{-1} A_{.BV}) x_{BV} + (A_{.BV}{}^{-1} A_{.Q}) x_Q = A_{.BV}{}^{-1} b$, $x_{BV} \geq 0$, and $x_Q \geq 0$. Let $BV^*$ be an optimal basis to MPLP and $BV^*$ replaces $BV$. This process continues until an optimal basis to SLP is obtained. Observe that if a non-optimal basis is selected to MPLP, then $BV \cup Q$ could remain unchanged, and the algorithm may never terminate. Consequently, either Shamos and Hoey (1976), Megiddo (1983), or Dyer (1984) algorithms cannot be directly applied to solve the MPLP when $|Q| = 2$ since none of these methods determine the optimal basis.

In order to view SM in terms of a multiple pivot simplex method, observe that $|Q| = 1$ and MPLP consists of $r + 1$ variables, $r$ constraints, and $r + 1$ nonnegativity constraints. The $r + 1$ variables are the $r$ basic variables and the one entering nonbasic variable. The ratio test identifies the optimal basis of this problem. Therefore, a classic pivot identifies the optimal basis of an MPLP with $r + 1$ variables.

The next section describes the double pivot simplex method (DPSM), which is a novel multiple pivot simplex method. DPSM has $|Q| = 2$ and each subproblem is optimized using SA. Even though other researchers have created multiple pivot simplex methods with $|Q| > 2$, these researchers used classic pivots to obtain the optimal basis to their subproblems (Howard 1960; Padberg 1999; Ye 2011). Thus, DPSM is novel and could be used to solve the subproblems from these other multiple pivot methods as well.

## 4.1 The double pivot simplex method

The first step in generating the double pivot simplex method (DPSM) is to prove that SA identifies an optimal basis for any MPLP with $|Q| = 2$. To prove this claim, convert S2LP into standard form (SS2LP) by adding $r$ slack variables. Each constraint of SS2LP has the form of $a_{j,1}x_1 + a_{j,2}x_2 + x_{j+2} = b_j$ for all $j \in R$. If implemented within a simplex method environment, these slack variables are precisely the existing basic variables. Furthermore, the only nonzero reduced costs are $c_1$ and $c_2$, which are both positive. The right-hand side is precisely the values of the basic variables, which are greater than or equal to zero. Removing the basic variables from this instance results in S2LP. Theorem 4 proves that SA returns sufficient information to determine an optimal basis, assuming that SS2LP is bounded. In addition, this result also provides an alternate proof to Theorem 3 in which SA identifies an optimal solution to S2LP.

**Theorem 4** *Implementing SA on a bounded S2LP with output $j^*$, $k^*$, $\alpha_{j^*}$, and $\alpha_{k^*}$ results in the following sets being an optimal basis for SS2LP:*

(i) *If $\alpha_{j^*} = -2M$, then $BV = \{3, 4, \ldots, k^* + 1, 1, k^* + 3, \ldots, r + 2\}$;*
(ii) *If $\alpha_{k^*} = 2M$, then $BV = \{3, 4, \ldots, j^* + 1, 2, j^* + 3, \ldots, r + 2\}$;*
(iii) *If $\alpha_{j^*} \neq -2M$ and $\alpha_{k^*} \neq 2M$, then $BV = \{3, 4, \ldots, j^* + 1, 1, j^* + 3, \ldots, k^* + 1, 2, k^* + 3, \ldots, r + 2\}$.*

*Proof* Implementing SA on a bounded S2LP returns $j^*$, $k^*$, $\alpha_{j^*}$, and $\alpha_{k^*}$. In order to prove $BV$ is an optimal basis to SS2LP, all possible cases of $\alpha_{j^*}$ and $\alpha_{k^*}$ are examined.

Assume $\alpha_{j^*} = -2M$, and let $BV = \{3, 4, \ldots, k^* + 1, 1, k^* + 3, \ldots, r + 2\}$ with $NBV = \{2, k^* + 2\}$. Since S2LP is bounded, $-M < \alpha_{k^*} < M$, $a_{k^*,1} > 0$, and $a_{k^*,2} > 0$; therefore, $c^{\pi}_{k^*+2} = \frac{c_1}{a_{k^*,1}} > 0$ and $c^{\pi}_2 = \frac{a_{k^*,2}c_1}{a_{k^*,1}} - c_2$, where $c^{\pi}$ is the calculated reduced cost. According to SA, $\frac{c_2}{c_1} \leq \alpha_{k^*}$ and the conditions of Corollary 2 are satisfied. Therefore, $c_2 a_{k^*,1} \leq c_1 a_{k^*,2}$, which implies $c^{\pi}_2 \geq 0$. Since $a_{k^*,1} > 0$, the columns of $BV = \{3, 4, \ldots, k^* + 1, 1, k^* + 3, \ldots, r + 2\}$ in SS2LP are linearly independent and therefore, $BV$ is an optimal basis to SS2LP.

Assume $\alpha_{k^*} = 2M$, $BV = \{3, 4, \ldots, j^* + 1, 2, j^* + 3, \ldots, r + 2\}$, and $NBV = \{1, j^* + 2\}$. Since SS2LP is bounded, $-M < \alpha_{j^*} < M$, $a_{j^*,1} > 0$, and $a_{j^*,2} > 0$; therefore, $c^{\pi}_{j^*+2} = \frac{c_2}{a_{j^*,2}} > 0$ and $c^{\pi}_1 = \frac{a_{j^*,1}c_2}{a_{j^*,2}} - c_1$. According to SA, $\alpha_{j^*} < \frac{c_2}{c_1}$ and the conditions of Corollary 1 are satisfied. Therefore, $c_1 a_{j^*,2} < c_2 a_{j^*,1}$, which implies $c^{\pi}_1 > 0$. Since $a_{j^*,2} > 0$, the columns of $BV = \{3, 4, \ldots, j^*+1, 2, j^*+3, \ldots, r+2\}$ in SS2LP are linearly independent, and $BV$ is an optimal basis to SS2LP.

Assume that $\alpha_{j^*} \neq -2M, \alpha_{k^*} \neq 2M$, $BV = \{3, 4, \ldots, j^*+1, 1, j^*+3, \ldots, k^*+1, 2, k^* + 3, \ldots, r + 2\}$, and $NBV = \{j^* + 2, k^* + 2\}$. The first step is to prove that the columns of $BV$ in SS2LP are linearly independent. Since $\alpha_{j^*} < \frac{c_2}{c_1} \leq \alpha_{k^*} \leq 2M$, the conditions of Lemma 2 are satisfied. Thus, $a_{j^*,2}a_{k^*,1} < a_{j^*,1}a_{k^*,2}$ and so $a_{j^*,2}a_{k^*,1} - a_{j^*,1}a_{k^*,2} < 0$ (†). Consequently, the columns of $BV$ in SS2LP are linearly independent. One can verify that $c^{\pi}_{k^*+2} = \frac{-c_2 a_{j^*,1} + c_1 a_{j^*,2}}{a_{j^*,2}a_{k^*,1} - a_{j^*,1}a_{k^*,2}}$. The conditions of Corollary 1 are satisfied, so $a_{j^*,2}c_1 < a_{j^*,1}c_2$. Combining this fact with (†) results in $c^{\pi}_{k^*+2} > 0$. Similarly, $c^{\pi}_{j^*+2} = \frac{c_2 a_{k^*,1} - c_1 a_{k^*,2}}{a_{j^*,2}a_{k^*,1} - a_{j^*,1}a_{k^*,2}}$. The conditions of Corollary 2 are satisfied, so $c_2 a_{k^*,1} \leq c_1 a_{k^*,2}$. Coupling this fact with (†) results in $c^{\pi}_{j^*+2} \geq 0$. Thus, $BV$ is an optimal basis for SS2LP. □

With the primary results, DPSM (Algorithm 2) is presented within the context of a revised simplex framework. The reader can easily create a dictionary, or tableau version. Even though not necessary, DPSM follows the spirit of Dantzig's rule and selects the two indices with the most negative reduced cost for the entering variables. DPSM's input is an SLP, a feasible basis $BV$ (typically the slack variables), and a sufficiently large number $M$. Observe that DPSM performs a classic pivot if there is only one negative reduced cost (lines 14–22).

In the absence of degeneracy (Sect. 4.2), DPSM optimally solves an LP following an identical argument as the proof of correctness for SM. If the current solution has two or more variables with negative reduced costs, then DPSM performs a double pivot. If the current solution has only one variable with negative reduced cost, then DPSM selects this variable and performs a classic pivot. Therefore, either pivot results in an updated basis with an improved objective value (considering a nondegenerate problem). Since there are a finite number of bases, DPSM either finds a ray of unboundedness or the optimal basis, which identifies the optimal solution.

In order to assess the benefit of DPSM, one should compare the improvement and effort per iteration of SM and DPSM. Assume both methods have the same basic feasible variables. If one of DPSM's entering variables is identical to the entering variable, $x_p$, in SM, then the objective value from DPSM's pivot is at least as good as

**Algorithm 2** - The double pivot simplex method (DPSM)

```
1: begin
2:   while $c_{BV}{}^T A_{.BV}{}^{-1} A - c \ngeq 0$ do
3:     $p \leftarrow \underset{p \in N \setminus BV}{\text{argmin}} \; c_{BV}^T A_{.BV}{}^{-1} A_{.p} - c_p$;
4:     $q \leftarrow \underset{q \in N \setminus (BV \cup \{p\})}{\text{argmin}} \; c_{BV}^T A_{.BV}{}^{-1} A_{.q} - c_q$;
5:     if $c_{BV}{}^T A_{.BV}{}^{-1} A_{.q} - c_q < 0$ then
6:       Let S2LP be: Maximize $z = -(c_{BV}{}^T A_{.BV}{}^{-1} A_{.p} - c_p)x_p - (c_{BV}{}^T A_{.BV}{}^{-1} A_{.q} - c_q)x_q$
                    Subject to $(A_{.BV}{}^{-1} A_{.p})x_p + (A_{.BV}{}^{-1} A_{.q})x_q \leq A_{.BV}{}^{-1} b$
                               $x_p \geq 0$ and $x_q \geq 0$;
7:       Solve S2LP with SA;
8:       if S2LP is unbounded then return SLP is unbounded;
9:       if $\alpha_{j^*} = -2M$ then $BV_{k^*} \leftarrow p$;
10:      if $\alpha_{k^*} = 2M$ then $BV_{j^*} \leftarrow q$;
11:      if $\alpha_{j^*} \neq -2M$ and $\alpha_{k^*} \neq 2M$ then
12:        $BV_{j^*} \leftarrow p$;
13:        $BV_{k^*} \leftarrow q$;
14:    else
15:      $\theta \leftarrow M$;
16:      for each $i \in R$ do
17:        if $(A_{.BV}{}^{-1} A_{.p})_i > 0$ and $\frac{(A_{.BV}{}^{-1} b)_i}{(A_{.BV}{}^{-1} A_{.p})_i} < \theta$ then
18:          $\theta \leftarrow \frac{(A_{.BV}{}^{-1} b)_i}{(A_{.BV}{}^{-1} A_{.p})_i}$;
19:          $l \leftarrow i$;
20:      if $\theta = M$ then return SLP is unbounded;
21:      else
22:        $BV_l \leftarrow p$;
23:   return $x_{BV}^* \leftarrow A_{.BV}{}^{-1} b$, $x_{N \setminus BV}^* = 0$, and $z^* = c^T x^*$;
24: end
```

the objective value from SM's pivot. Furthermore, SM and DPSM only pivot to the same basis when SA returns $\alpha_{j^*} = -2M$.

The theoretical effort required by an iteration of SM with a classic pivot involves calculating $A_{.BV}{}^{-1}$ and identifying an improving variable $x_p$, which is achieved by evaluating $c_{BV}{}^T A_{.BV}{}^{-1} A_{.NBV} - c_{NBV}$. The ratio test calculates $A_{.BV}{}^{-1} A_{.p}$, $A_{.BV}{}^{-1} b$, and performs a division. Changing the entering variable with the leaving variable requires $O(1)$ effort. Numerous methods have been developed to decrease the computational effort required to perform each of these tasks and the fastest running times are dependent upon the implemented improvements. For example, finding the inverse can be accomplished in $O(r^3)$ (Edmonds 1967; Schrijver 1998), but there are faster methods to merely update an inverse (Strassen 1969; Coppersmith and Winograd 1990; Williams 2012). Thus, the effort per iteration is limited by calculating the inverse and identifying the entering and leaving variables. Consequently, each iteration of SM requires $O(rn + I(r))$ effort where $I(r)$ is the time required to find the inverse of an $r \times r$ matrix.

The theoretical effort for an iteration of DPSM requires calculating $A_{.BV}{}^{-1}$ and identifying two improving variables, $x_p$ and $x_q$. These steps are nearly identical to SM and require identical theoretical effort. SA determines the leaving variables in $O(S(r))$

effort, where $S(r)$ is the time required to sort a set of $r$ elements. Exchanging the basic variables is performed again in $O(1)$. Thus, each double pivot is restricted by calculating the inverse and identifying the entering variables, which requires $O(rn + I(r))$ effort. Consequently, a double pivot and a classic pivot require the same theoretical effort per iteration.

In practice, commercial or open source solvers do not calculate the inverse at every iteration, instead solvers update the basis. Numerous researchers devised schemes to efficiently update the basis matrix (Dantzig and Orchard-Hays 1954; Bartels 1971; Forrest and Tomlin 1972; Reid 1982; Eldersveld and Saunders 1992; Suhl and Suhl 1993; Huangfu and Julian Hall 2015). Consequently, the theoretical run time of SM in practice is $O(rn + U(r))$ where $U(r)$ is the effort required to update the basis matrix of a problem with $r$ constraints. Since $n \geq r$ due to the addition of slack or artificial variables, SA's run time is dominated. Thus, DPSM also requires $O(rn + U(r))$ effort in practice. Example 2 demonstrates DPSM in a tableau format.

*Example 2* Consider the following LP.

$$\begin{array}{lll}
\text{Maximize } z = 20x_1 + 12x_2 + 15x_3 + 6x_4 & & \\
\text{Subject to} \quad x_1 - 2x_2 + 3x_3 + x_4 \leq 99 & & (1) \\
\quad x_1 + x_3 \leq 40 & & (2) \\
\quad 4x_1 + 9x_2 + x_3 + 4x_4 \leq 106 & & (3) \\
\quad 2x_1 + 2x_2 + x_3 + x_4 \leq 60 & & (4) \\
\quad 2x_1 - x_2 + 5x_3 \leq 170 & & (5) \\
\quad x_1, x_2, x_3, x_4 \geq 0 & &
\end{array}$$

Table 1 contains three tableaus that demonstrate DPSM's pivots. The first tableau represents the above LP in standard form and DPSM begins with $BV = \{5, 6, 7, 8, 9\}$. The variables with the two most negative reduced costs correspond to $x_1$ and $x_3$ with $p = 1$ and $q = 3$. SA solves maximize $z = 20x_1 + 15x_3$ subject to $A_{.\{1,3\}}(x_1, x_3)^T \leq b$, $x_1 \geq 0$, and $x_3 \geq 0$. SA calculates $\alpha = (3, 1, \frac{1}{4}, \frac{1}{2}, \frac{5}{2}, 2M, -2M)$ and identifies $\alpha_4 = \frac{1}{2} < \frac{c_3}{c_1} = \frac{15}{20} \leq \alpha_2 = 1$ with $\overline{x} = (x_1, x_3) = (20, 20)$. This point satisfies all constraints, and SA returns $j^* = 4$, $k^* = 2$, $\alpha_4 = \frac{1}{2}$, and $\alpha_2 = 1$. Due to the returned values, the nonbasic indices 1 and 3 replace the fourth and second elements in $BV$, resulting in $BV = \{5, 3, 7, 1, 9\}$. The second tableau in Table 1 demonstrates this double pivot's outcome.

The next iteration begins by identifying the variables with the two most negative reduced costs, $x_2$ and $x_4$, resulting in $p = 2$ and $q = 4$. SA solves maximize $z = 2x_2 + x_4$ subject to $A^{-1}_{.BV} A_{.\{2,4\}}(x_2, x_4)^T \leq A^{-1}_{.BV} b$, $x_2 \geq 0$, and $x_4 \geq 0$. SA calculates $\alpha = (\frac{3}{2}, 3M, \frac{1}{3}, \frac{1}{2}, \frac{3}{5}, 2M, -2M)$, identifies $\alpha_3 = \frac{1}{3} < \frac{c_4}{c_2} = \frac{1}{2} \leq \alpha_4 = \frac{1}{2}$, and assigns $\overline{x}$ to $(-14, 48)$. SA eventually determines that the optimal solution occurs at $x^* = (0, 6)$, the intersection point of (3) and (6), and returns $j^* = 3$, $k^* = 6$, $\alpha_3 = \frac{1}{3}$, and $\alpha_6 = 2M$. Since $\alpha_{k^*} = 2M$, the nonbasic index 4 replaces the third element in $BV$, resulting in $BV = \{5, 3, 4, 1, 9\}$. The third tableau in Table 1 presents the result of this double pivot. Observe that there are no variables with negative reduced cost in the third tableau. Thus, $BV = \{5, 3, 4, 1, 9\}$ is an optimal basis to the SLP and DPSM reports the optimal solution $z^* = 706$ and $x^* = (14, 0, 26, 6, 1, 0, 0, 0, 12)$.

**Table 1** Double pivot simplex tableau—Example 2

| BV | $z$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ | RHS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | − 20 | − 12 | −15 | − 6 | 0 | 0 | 0 | 0 | 0 | 0 |
| $x_5$ | 0 | 1 | − 2 | 3 | 1 | 1 | 0 | 0 | 0 | 0 | 99 |
| $x_6$ | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 40 |
| $x_7$ | 0 | 4 | 9 | 1 | 4 | 0 | 0 | 1 | 0 | 0 | 106 |
| $x_8$ | 0 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 60 |
| $x_9$ | 0 | 2 | − 1 | 5 | 0 | 0 | 0 | 0 | 0 | 1 | 170 |
| | 1 | 0 | −2 | 0 | −1 | 0 | 10 | 0 | 5 | 0 | 700 |
| $x_5$ | 0 | 0 | 2 | 0 | 3 | 1 | − 5 | 0 | 2 | 0 | 19 |
| $x_3$ | 0 | 0 | − 2 | 1 | −1 | 0 | 2 | 0 | − 1 | 0 | 20 |
| $x_7$ | 0 | 0 | 3 | 0 | 1 | 0 | 2 | 1 | − 3 | 0 | 6 |
| $x_1$ | 0 | 1 | 2 | 0 | 1 | 0 | − 1 | 0 | 1 | 0 | 20 |
| $x_9$ | 0 | 0 | 5 | 0 | 3 | 0 | − 8 | 0 | 3 | 1 | 30 |
| | 1 | 0 | 1 | 0 | 0 | 0 | 12 | 1 | 2 | 0 | 706 |
| $x_5$ | 0 | 0 | − 7 | 0 | 0 | 1 | − 11 | −3 | 11 | 0 | 1 |
| $x_3$ | 0 | 0 | 1 | 1 | 0 | 0 | 4 | 1 | − 4 | 0 | 26 |
| $x_4$ | 0 | 0 | 3 | 0 | 1 | 0 | 2 | 1 | − 3 | 0 | 6 |
| $x_1$ | 0 | 1 | − 1 | 0 | 0 | 0 | − 3 | −1 | 4 | 0 | 14 |
| $x_9$ | 0 | 0 | − 4 | 0 | 0 | 0 | − 14 | −3 | 12 | 1 | 12 |

When comparing SM to DPSM, SM solves the LP from Example 2 with four iterations. Starting with $BV = \{5, 6, 7, 8, 9\}$, SM moves to the following bases: $\{5, 6, 1, 8, 9\}$, $\{5, 6, 1, 3, 9\}$, $\{5, 2, 1, 3, 9\}$, and $\{5, 4, 1, 3, 9\}$. Consequently, DPSM performs 50% fewer iterations than SM.

Observe that DPSM has three types of double pivots and this example presents two of them. The first double pivot replaces two variables in the basis. The second double pivot exchanges only one variable in the basis, which implies that one of the entering variables, $x_2$, is also a "leaving" variable. In this pivot, the index that enters the basis corresponds to the variable with the second most negative reduced cost, $x_4$. The other type of double pivot ($\alpha_{j^*} = -2M$) corresponds exactly to a classic pivot and the variable with the most negative reduced cost is the only entering variable.

Double pivots are guaranteed to improve the objective function by at least as much as classic pivots. The classic pivot from the starting basis results in an objective value of $z = 530$. Thus, the relative improvement of a double pivot is $\left(\frac{700}{530} - 1\right) \times 100\% = 32.1\%$. Performing a classic pivot from $BV = \{5, 3, 7, 1, 9\}$ results in an objective value of $z = 704$. The second double pivot's relative improvement is $\left(\frac{706-700}{704-700} - 1\right) \times 100\% = 50\%$. Thus, even if a double pivot has a single entering and leaving variable, the benefit may be substantial.

### 4.2 The double pivot simplex method and degeneracy

Degeneracy causes issues for SM and occurs when at least one basic variable of an SLP equals zero. Degenerate SLPs may force SM to complete extra operations by performing multiple iterations at the same feasible solution. Additionally, degenerate SLPs may cycle, which prohibits SM from terminating. Implementing anti-cycling techniques enables SM to avoid cycling and terminate (Bland 1977; Todd 1985; He 1999; Elhallaoui et al. 2010; Raymond et al. 2010). This section demonstrates that implementing SA instead of the ratio test in a simplex framework diminishes some of the issues caused by degeneracy.

Theorem 3 guarantees that an optimal solution to a bounded SA2LP occurs at the intersection point of the $j$th and $k$th constraints as long as this point is feasible, $\alpha_j < \frac{c_2}{c_1}$, $\alpha_k \geq \frac{c_2}{c_1}$, and SA2LP$_{j,k}$ is bounded. SA returns constraints $j^*$ and $k^*$, which not only fulfill Theorem 3's conditions, but also satisfy $\alpha_{k^*} - \alpha_{j^*} \leq \alpha_k - \alpha_j$ for all $j$ and $k$ pairs of constraints that meet Theorem 3's conditions (Algorithm 1, line 5). The selection of these particular constraints results in an optimal basis according to Theorem 4. Example 3 provides a degenerate problem that helps explain this concept.

*Example 3* Consider the following S2LP.

$$
\begin{array}{lll}
\text{Maximize } z = 5x_1 + 4x_2 & & \\
\text{Subject to} & x_1 - x_2 \leq 3 & (1) \\
 & x_1 \leq 3 & (2) \\
 & 2x_1 + x_2 \leq 6 & (3) \\
 & 3x_1 + 2x_2 \leq 10 & (4) \\
 & x_1 + x_2 \leq 4 & (5) \\
 & 2x_1 + 3x_2 \leq 10 & (6) \\
 & x_1 + 2x_2 \leq 6 & (7) \\
 & x_2 \leq 3 & (8) \\
 & -x_1 + x_2 \leq 3 & (9) \\
 & -x_1 - x_2 \leq 0 & (10) \\
 & x_1, x_2 \geq 0 &
\end{array}
$$

The graphical representation of the S2LP is presented in Fig. 4. Solving this S2LP with SA results in $z^* = 18, x^* = (2, 2), j^* = 4, k^* = 5$. Observe that $\alpha_4 = \frac{2}{3} < \frac{c_2}{c_1} = \frac{4}{5} < \alpha_5 = 1$ and $\alpha_5 - \alpha_4 = \frac{1}{3}$. There are six pairs of constraints that satisfy Theorem 3 and identify the optimal solution: (3) and (5), (3) and (6), (3) and (7), (4) and (5), (4) and (6), and (4) and (7). The reader can verify that none of the bases corresponding to these pairs of constraints are optimal unless the basis is derived from (4) and (5). Therefore, SA identifies the optimal basis even for degenerate S2LPs. Observe that the algorithms by Shamos and Hoey (1976), Megiddo (1983), or Dyer (1984) all solve two variable LPs. However, these algorithms may end with any one of these six pairs of constraints intersecting at the optimal solution. Consequently, these fast algorithms do not always identify the optimal basis and cannot be used as a multiple pivoting strategy within a simplex framework as previously mentioned.

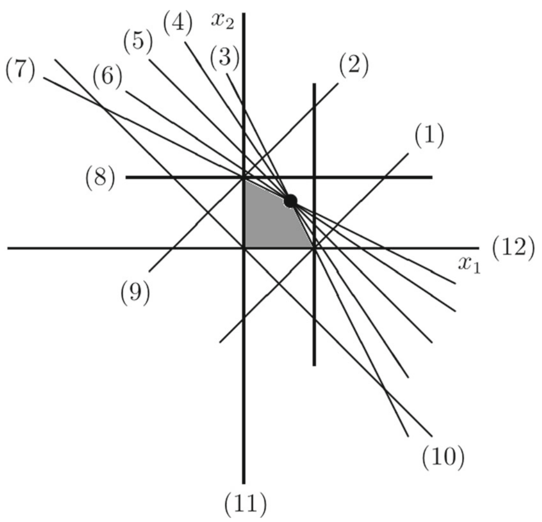


**Fig. 4** Graphical representation of Example 3

In contrast, implementing classic pivots on this degenerate S2LP demonstrates a weakness of SM. Performing four classic pivots with Bland's rule results in a basis with constraints (5) and (6). Even though this basis identifies the optimal solution, one more pivot is required to obtain the optimal basis. Thus, degeneracy caused additional work for SM, but not for DPSM.

In order to determine whether or not DPSM performs better than SM on degenerate problems that cycle, consider the 11 instances summarized by Gass and Vinjamuri (2004). These 11 problems cycle when solved with SM without any anti-cycling technique. However, DPSM identifies the optimal solution (or an unbounded LP) in every one of these 11 LPs. Therefore, DPSM avoids cycling on these frequently demonstrated degenerate LPs.

In conclusion, DPSM handles the problems caused by degenerate LPs more effectively than SM. One should not infer from this section that DPSM completely eliminates all issues of degenerate LPs, and determining whether or not DPSM cycles is an important unresolved research question.

## 5 Computational study

This section discusses the authors' attempts to determine whether or not DPSM is computationally faster than SM. The study was performed on an Intel® Core$^{TM}$ i7-6700 3.4 GHz processor with 32 GB of RAM, and coded in C++. A portion of the study implements DPSM and SM with CPLEX Version 12.5, a high performance mathematical programming solver. CPLEX's preprocessing operations were turned off in order to measure the true effectiveness of DPSM over SM.

## 5.1 Implementation of DPSM and SM

The authors began the study by implementing DPSM and SM explicitly. The code not only computes ${A_{.BV}}^{-1}$ at every iteration using LU decomposition (Press et al. 2007), but determines explicitly the reduced cost ${c_{BV}}^T {A_{.BV}}^{-1} A - c$ of all nonbasic variables, the right-hand side $A^{-1}_{.BV} b$, and the constraint values $A^{-1}_{.BV} A_{.p}$ and $A^{-1}_{.BV} A_{.q}$ of both improving variables.

This implementation was tested on 3000 randomly generated instances and DPSM outperformed SM with an average improvement in solution time of approximately 17% for dense instances and 30% for sparse problems. Furthermore, DPSM also had an average of approximately 17% fewer pivots for dense instances and 30% for sparse problems. The number of pivots was highly correlated to the solution time. These instances were also run with CPLEX's primal algorithm. CPLEX surpassed both methods and solved the instances in a few seconds, while both DPSM and SM required hours. This result is not surprising as explicitly implementing DPSM or SM eliminates decades of computational advancements.

To attempt to take advantage of these computational advancements, the authors implemented DPSM and SM using many of the routines available in CPLEX. Entering variable for SM is obtained from the *CPXgetdj* routine. Routines *CPXgetx* and *CPXbinvacol* are used to perform the ratio test and determine the leaving variable. The *CPXpivot* routine updates the basis by swapping the leaving variable with the entering variable.

A similar implementation is followed for DPSM. Both entering variables are obtained from the *CPXgetdj* routine. Two calls to the *CPXbinvacol* routine and one call to the *CPXgetx* routine creates an S2LP. SA solves the S2LP and identifies the leaving and entering variable(s). Unfortunately, the *CPXpivot* routine only allows for a single exchange of variables. Thus, if two variables enter the basis, the *CPXpivot* routine is called twice. If either of the other type of pivots occur, the *CPXpivot* routine is called once.

Both DPSM and SM require a starting feasible basis, which can be obtained from a Phase 1 implementation using either DPSM or SM. In this study, CPLEX's Phase 1 reported a feasible basis, which is used as the starting basis for both DPSM and SM.

Solution times obtained with these implementations became comparable with CPLEX's primal algorithm. Obviously, these implementations are still slower, but it only slowed the solution time by less than 50%. With a reasonable implementation of DPSM and SM, the study solved benchmark problems from Netlib (Gay 1985) and MIPLIB (Koch et al. 2011). Instances from MIPLIB are mixed integer programs and were changed to LPs by eliminating the integrality constraints. To avoid the upper and lower bound simplex implementations, additional constraints were added to the problems in order to represent variables with lower and upper bounds.

During the computational experiments, the authors encountered serious issues with numerical instability from the *CPXpivot* routine. This issue prohibited DPSM and SM from terminating and/or led to numerically singular bases for many of the benchmark instances. When a basis is found to be singular, CPLEX removes one or more variables from the current basis and re-includes these variables on further iterations when an optimal basis is obtained. If after re-including these variables the basis is no longer optimal, then CPLEX proceeds until an optimal basis is found; otherwise, an optimal solution to the problem has been found.

Unfortunately, the authors could not duplicate CPLEX's internal repair process in the implementation of SM or DPSM. Failure to correctly repair the basis may result in incorrect and/or illegal pivots. The authors attempted to fix this problem through more frequent refactoring of the basis (*CPX_PARAM_REINV*), restricting the number of times CPLEX repairs the basis (*CPX_PARAM_SINGLIM*), and tracking the kappa value to identify when a pivot makes the basis become unstable. However, none of these attempts resolved the issue.

The reader should know that CPLEX's numerical instability occurred in both DPSM and SM. The computational experiments found problems where DPSM solved, but SM did not, and vice versa. Therefore, this computational study only reports the results of the benchmark problems where both DPSM and SM solved the instance.

## 5.2 Computational results and analysis

Tables 2 and 3 describe all instances solved from Netlib and MIPLIB, respectively. These tables present the number of pivots (Phase 2 only) performed by each method, including CPLEX's primal algorithm. It also describes the percentage improvement obtained with DPSM over SM and DPSM over CPLEX. Improvement is defined as $\delta_{\frac{DPSM}{SM}} = \frac{y_{SM} - y_{DPSM}}{y_{SM}} \times 100\%$ and $\delta_{\frac{DPSM}{CPLEX}} = \frac{y_{CPLEX} - y_{DPSM}}{y_{CPLEX}} \times 100\%$ where $y_{DPSM}$ is the number of pivots performed by DPSM, $y_{SM}$ indicates the number of pivots performed by SM, and $y_{CPLEX}$ equals the number of pivots performed by CPLEX's primal algorithm.

Tables 2 and 3 show that DPSM averages 41% fewer pivots than SM and 23% fewer pivots than CPLEX's primal algorithm. Each double pivot identified an average of only 2 constraints that updated $\overline{x}$, implying that Theorem 2 and Corollaries 3, 4, and 5 are infrequently implemented. The study shows that on average 83% of pivots entered two variables in the basis, 2% entered the variable with the most negative reduced cost, and 15% entered the variable with the second most negative reduced cost. Overall, only 0.05% of the iterations had a single negative reduced cost, which implies that the vast majority of the iterations pivoted with SA and not with the ratio test.

The average objective's relative improvement per pivot of all benchmark instances solved is $\overline{\delta} = 170\%$. Define the objective's improvement per iteration as $\delta = \left(\frac{z_{double\ pivot} - z_{current}}{z_{classic} - z_{current}} - 1\right) \times 100\%$ where $z_{current}$ is the $z$ value of the current basis, $z_{double\ pivot}$ is the $z$ value after a double pivot, and $z_{classic}$ is the $z$ value after a classic pivot. If a classic pivot does not improve the objective function, then $\delta = \left(\frac{z_{double\ pivot} - z_{current}}{z_{current}}\right) \times 100\%$. In other words, $\delta$ represents the percent in objective improvement that occurs by using a double pivot compared to the improvement attributed to the use of a classic pivot. Thus, all values of $\delta$ are averaged to create $\overline{\delta}$.

When analyzing based on pivot type, $\overline{\delta} = 473\%$ for double pivots that enter two nonbasic variables into the basis, $\overline{\delta} = 135\%$ for double pivots that enter the variable with the second most negative reduced cost, and obviously $\overline{\delta} = 0\%$ for double pivots that enter the variable with the most negative reduced cost.

**Table 2** Results of instances from Netlib

| Name | Rows | Columns | Nonzeros | Number of Pivots | | | $\delta_{\frac{DPSM}{SM}}$ (%) | $\delta_{\frac{DPSM}{CPLEX}}$ (%) |
|---|---|---|---|---|---|---|---|---|
| | | | | CPLEX | SM | DPSM | | |
| ADLITTLE | 57 | 97 | 465 | 58 | 100 | 56 | 44.0 | 3.4 |
| AGG | 489 | 163 | 2541 | 64 | 85 | 45 | 47.1 | 29.7 |
| AGG2 | 517 | 302 | 4515 | 85 | 87 | 55 | 36.8 | 35.3 |
| BANDM | 306 | 472 | 2659 | 227 | 305 | 152 | 50.2 | 33.0 |
| BEACONFD | 174 | 262 | 3476 | 88 | 95 | 52 | 45.3 | 40.9 |
| BRANDY | 221 | 249 | 2150 | 194 | 271 | 208 | 23.2 | −7.2 |
| CZPROB | 930 | 3523 | 14173 | 1122 | 3741 | 1681 | 55.1 | −49.8 |
| DEGEN2 | 445 | 534 | 4449 | 1041 | 6771 | 1145 | 83.1 | −10.0 |
| FIT1D | 25 | 1026 | 14430 | 911 | 1064 | 847 | 20.4 | 7.0 |
| FIT1P | 628 | 1677 | 10894 | 987 | 1357 | 797 | 41.3 | 19.3 |
| FIT2D | 26 | 10500 | 138018 | 20166 | 13298 | 11663 | 12.3 | 42.2 |
| GANGES | 1310 | 1681 | 7021 | 294 | 678 | 268 | 60.5 | 8.8 |
| GROW15 | 301 | 645 | 5665 | 977 | 871 | 759 | 12.9 | 22.3 |
| GROW7 | 141 | 301 | 2633 | 208 | 246 | 139 | 43.5 | 33.2 |
| KB2 | 44 | 41 | 291 | 35 | 40 | 37 | 7.5 | −5.7 |
| LOTFI | 154 | 308 | 1086 | 177 | 377 | 261 | 30.8 | −47.5 |
| RECIPELP | 92 | 180 | 752 | 33 | 34 | 16 | 52.9 | 51.5 |
| SC50A | 51 | 48 | 131 | 20 | 18 | 11 | 38.9 | 45.0 |
| SC50B | 51 | 48 | 119 | 19 | 16 | 11 | 31.3 | 42.1 |
| SCAGR25 | 472 | 500 | 2029 | 357 | 433 | 259 | 40.2 | 27.5 |
| SCAGR7 | 130 | 140 | 553 | 81 | 84 | 52 | 38.1 | 35.8 |
| SCFXM1 | 331 | 457 | 2612 | 171 | 227 | 118 | 48.0 | 31.0 |
| SCFXM2 | 661 | 914 | 5229 | 327 | 503 | 276 | 45.1 | 15.6 |
| SCFXM3 | 991 | 1371 | 7846 | 502 | 794 | 445 | 44.0 | 11.4 |
| SCORPION | 389 | 358 | 1708 | 270 | 271 | 242 | 10.7 | 10.4 |
| SHARE1B | 118 | 225 | 1182 | 150 | 404 | 198 | 51.0 | −32.0 |
| SHARE2B | 97 | 79 | 730 | 40 | 80 | 39 | 51.3 | 2.5 |
| SHELL | 537 | 1775 | 4900 | 325 | 404 | 244 | 39.6 | 24.9 |
| SHIP04L | 403 | 2118 | 8450 | 346 | 366 | 195 | 46.7 | 43.6 |
| SHIP04S | 403 | 1458 | 5810 | 155 | 145 | 76 | 47.6 | 51.0 |
| SHIP08L | 779 | 4283 | 17085 | 668 | 717 | 382 | 46.7 | 42.8 |
| SHIP08S | 779 | 2387 | 9501 | 454 | 567 | 291 | 48.7 | 35.9 |
| SHIP12L | 1152 | 5427 | 21597 | 276 | 262 | 153 | 41.6 | 44.6 |
| SHIP12S | 1152 | 2763 | 10941 | 187 | 187 | 101 | 46.0 | 46.0 |
| STANDATA | 360 | 1075 | 3038 | 138 | 113 | 62 | 45.1 | 55.1 |
| STANDMPS | 468 | 1075 | 3686 | 418 | 409 | 245 | 40.1 | 41.4 |
| STOCFOR1 | 118 | 111 | 474 | 91 | 59 | 32 | 45.8 | 64.8 |
| STOCFOR3 | 16676 | 15695 | 74004 | 18931 | 21221 | 18254 | 14.0 | 3.6 |
| Average | | | | | | | **40.2** | **22.4** |

**Table 3** Results of instances from MIPLIB

| Name | Rows | Columns | Nonzeros | Number of Pivots | | | $\delta_{\frac{DPSM}{SM}}$ (%) | $\delta_{\frac{DPSM}{CPLEX}}$ (%) |
|---|---|---|---|---|---|---|---|---|
| | | | | CPLEX | SM | DPSM | | |
| 50v-10 | 233 | 2013 | 2745 | 222 | 212 | 116 | 45.3 | 47.7 |
| dfn-gwin-UUM | 158 | 938 | 2632 | 137 | 126 | 76 | 39.7 | 44.5 |
| ger50_17_trans | 499 | 22414 | 172035 | 751 | 623 | 345 | 44.6 | 54.1 |
| germanrr | 10779 | 10813 | 175547 | 78 | 77 | 55 | 28.6 | 29.5 |
| ic97_potential | 1046 | 728 | 3138 | 47 | 48 | 24 | 50.0 | 48.9 |
| janos-us-DDM | 760 | 2184 | 6384 | 342 | 318 | 161 | 49.4 | 52.9 |
| mcsched | 2107 | 1747 | 8088 | 3118 | 4158 | 3666 | 11.8 | −17.6 |
| noswot | 182 | 128 | 735 | 10 | 33 | 10 | 69.7 | 0.0 |
| ns1766074 | 182 | 100 | 666 | 83 | 224 | 137 | 38.8 | −65.1 |
| timtab1 | 171 | 397 | 829 | 54 | 53 | 30 | 43.4 | 44.4 |
| Average | | | | | | | **42.1** | **23.9** |

Even though DPSM outperforms SM and CPLEX's primal algorithm in number of pivots, the question of whether or not DPSM is computationally faster still remains. Tables 2 and 3 do not include solution times because the vast majority of these instances were solved by DPSM, SM and CPLEX's primal algorithm in less than a tenth of a second. Obtaining reliable data on such small time increments is both inconclusive and unconvincing. It suffices to say that DPSM, SM, and CPLEX's primal algorithm were very close in computational time for these instances.

To provide a partial answer, some dense and sparse random LPs that did not present numerical instability issues with respect to the *CPXpivot* routine were solved with DPSM and SM. These LPs were large enough to produce reasonable solution times for comparison. In total, 50 problems that ranged from 2000 to 10,000 variables and 1000–5000 constraints were solved. On average, these problems solved with DPSM in 62 s, SM in 54 s, and CPLEX's primal algorithm in 39 s. When broken down by steps, SM spent on average 1% of the solution time to find the entering variable, 6% to obtain the right-hand side values and constraint matrix values of the entering variable, 1% to perform the ratio test, 91% to swap the leaving and entering variables using the *CPXpivot* routine, and 1% for all other operations. Similarly, DPSM spent on average 1% to find both entering variables, 11% to create S2LPs, 1% to find the leaving variable(s) with SA, 86% to exchange the leaving with entering variable(s) using the *CPXpivot* routine (called once or twice depending on the type of pivot), and 1% for all other operations. Therefore, the vast majority of time is spent pivoting and creating data for the ratio tests or S2LPs. Thus, SA is similar in computational speed to the ratio test, which follows the theoretical analysis that both algorithms have the same theoretical running time per iteration.

This small study demonstrates that SA does not significantly impact DPSM's solution time. The most expensive step corresponds to the *CPXpivot* routine. This routine not only updates the basis' inverse factorization, but also calculates the solution of all basic variables, reduced cost values, dual price values, etc. Unfortunately, DPSM frequently calls the *CPXpivot* routine twice during an iteration, which forces unnecessary work. Thus, this implementation of DPSM is at a competitive disadvantage compared to both SM and CPLEX's primal algorithm. Consequently, only a full implementation of DPSM in a quality commercial or open source linear programming solver can truly answer whether or not DPSM is faster than SM. Fully implementing DPSM requires the development of an efficient method to update the basis for double pivots.

## 6 Conclusions and future research

This paper introduces the double pivot simplex method, which improves one of the most famous and useful algorithms in science. At each iteration, the double pivot simplex method pivots on two variables, while the simplex method pivots on only one. The paper first presents the slope algorithm, a fast method to find the optimal basis and the optimal solution of a two variable linear program. Combining the slope algorithm within a simplex framework creates the double pivot simplex method. The slope algorithm is fast, and the most time-consuming step is sorting an array of numbers with size equal the number of constraints. This result enables an iteration of the double pivot simplex method to have the same theoretical running time as an iteration of the simplex method. Moreover, the objective value improvement per iteration of a double pivot is at least as large as the improvement from a classic pivot. In addition, the double pivot simplex method also diminishes some of the negative effects caused by degenerate linear programs. Computational experiments tested the double pivot simplex method on a small set of benchmark instances from Netlib and MIPLIB and showed that it reduces the number of pivots compared to the simplex method by over 40% on average.

The double pivot simplex method lays the foundation for a host of important future research topics. The first topic should determine the benefit of double pivots in state of the art linear programming solvers. The primary research task should develop an efficient method to update the basis factorization with two variables.

Another topic should extend double pivots to other simplex method results, including creating the double pivot dual simplex method, developing the double pivot simplex method with upper bounds, finding a small instance that cycles with the double pivot simplex method (or proving that the double pivot simplex method does not cycle), and generating pivoting rules to avoid cycling for the double pivot simplex method. Furthermore, the development of a triple (or more) pivot method is another promising research topic.

Research should also investigate how the double pivot simplex method can benefit from parallel computing. Since DPSM requires more work per iteration than SM (creating S2LP), can a portion of this step be performed in parallel? Along the same lines, if one answers the aforementioned triple or more pivot problem, a parallel implementation should become more effective.

Expanding the applicability of the slope algorithm generates additional future research topics. Currently, the majority of optimization algorithms move from one solution to another solution by following a single direction, which involves solving a one dimensional search problem. In contrast, the slope algorithm moves between solutions over a two dimensional space. Consequently, the slope algorithm is a two dimensional search algorithm. Can two dimensional search methods be developed to improve the solution time of interior point methods, nonlinear programming algorithms, and other optimization techniques?

## References

Alterovitz R, Lessard E, Pouliot J, Hsu I, O'Brien J, Goldberg K (2006) Optimization of HDR brachytherapy dose distributions using linear programming with penalty costs. Med Phys 33(11):4012–4019

Appelgren L (1969) A column generation algorithm for a ship scheduling problem. Transp Sci 3(1):53–68

Bartels R (1971) A stabilization of the simplex method. Numer Math 16(5):414–434

Bartolini F, Bazzani G, Gallerani V, Raggi M, Viaggi D (2007) The impact of water and agriculture policy scenarios on irrigated farming systems in Italy: an analysis based on farm level multi-attribute linear programming models. Agric Syst 93(1):90–114

Bazaraa M, Jarvis J, Sherali H (2009) Linear programming and network flows. Wiley, New Jersey

Bertsimas D, Tsitsiklis J (1997) Introduction to linear optimization. Athena Scientific, Belmont

Bland R (1977) New finite pivoting rules for the simplex method. Math Oper Res 2(2):103–107

Chalermkraivuth K, Bollapragada S, Clark M, Deaton J, Kiaer L, Murdzek J, Neeves W, Scholz B, Toledano D (2005) GE asset management, Genworth financial, and GE insurance use a sequential-linear-programming algorithm to optimize portfolios. Interfaces 35(5):370–380

Coppersmith D, Winograd S (1990) Matrix multiplication via arithmetic progressions. J Symb Comput 9(3):251–280

Dantzig G (1947) Maximization of a linear function of variables subject to linear inequalities. In: Koopmans TC (ed) Activity analysis of production and allocation, 1951. Wiley, New York, pp 339–347

Dantzig G (1982) Reminiscences about the origins of linear programming. Oper Res Lett 1(2):43–48

Dantzig G, Orchard-Hays W (1954) The product form for the inverse in the simplex method. Math Tables Aids Comput 8(46):64–67

Dantzig G, Wolfe P (1960) Decomposition principle for linear programs. Oper Res 8(1):101–111

Dongarra J, Sullivan F (2000) Guest editors' introduction: the top 10 algorithms. Comput Sci Eng 2(1):22–23

Dorfman R (1984) The discovery of linear programming. Ann Hist Comput 6(3):283–295

Dyer M (1984) Linear time algorithms for two- and three-variable linear programs. SIAM J Comput 13(1):31–45

Edmonds J (1967) Systems of distinct representatives and linear algebra. J Res Natl Bur Stand 71B(4):241–245

Eldersveld S, Saunders M (1992) A Block-LU update for large-scale linear programming. SIAM J Matrix Anal A 13(1):191–201

Elhallaoui I, Metrane A, Desaulniers G, Soumis F (2010) An improved primal simplex algorithm for degenerate linear programs. INFORMS J Comput 23(4):569–577

Ford L, Fulkerson D (1958) A suggested computation for maximal multi-commodity network flows. Manage Sci 5(1):97–101

Forrest J, Tomlin J (1972) Updated triangular factors of the basis to maintain sparsity in the product form simplex method. Math Program 2(1):263–278

García J, Florez J, Torralba A, Borrajo D, López C, García-Olaya Á, Sáenz J (2013) Combining linear programming and automated planning to solve intermodal transportation problems. Eur J Oper Res 227(1):216–226

Gass S, Vinjamuri S (2004) Cycling in linear programming problems. Comput Oper Res 31(2):303–311

Gautier A, Lamond B, Paré D, Rouleau F (2000) The québec ministry of natural resources uses linear programming to understand the wood-fiber market. Interfaces 30(6):32–48

Gay D (1985) Electronic mail distribution of linear programming test problems. Math Program Soc COAL Newslett 13:10–12

Gilmore P, Gomory R (1961) A linear programming approach to the cutting-stock problem. Oper Res 9(6):849–859

Gilmore P, Gomory R (1963) A linear programming approach to the cutting-stock problem—part II. Oper Res 11(6):863–888

Goldfarb D, Todd M (1989) Linear programming. In: Nemhauser GL, Rinnooy Kan AHG, Todd MJ (eds) Handbooks in operations research and management science, vol 1. North-Holland, Amsterdam, pp 73–170

Gomes A, Oliveira J (2006) Solving irregular strip packing problems by hybridising simulated annealing and linear programming. Eur J Oper Res 171(3):811–829

Gondzio J (2012) Interior point methods 25 years later. Eur J Oper Res 218(3):587–601

He J (1999) Homotopy perturbation technique. Comput Methods Appl Mech Eng 178(3–4):257–262

Hillier F, Lieberman G (2015) Introduction to operations research. McGraw-Hill, New York

Howard R (1960) Dynamic programming and Markov processes. The MIT Press, Cambridge

Huangfu Q, Julian Hall J (2015) Novel update techniques for the revised simplex method. Comput Optim Appl 60(3):587–608

Illés T, Terlaky T (2002) Pivot versus interior point methods: pros and cons. Eur J Oper Res 140(2):170–190

Kantorovich L (1939) Mathematical methods of organizing and planning production. Manage Sci 6(4):366–422 (1939 Russian, 1960 English)

Karmarkar N (1984) A new polynomial-time algorithm for linear programming. Combinatorica 4(4):373–395

Khachiyan L (1979) A polynomial algorithm in linear programming. Sov Math Dokl 20(1):191–194

Klee V, Minty G (1972) How good is the simplex algorithm? In: Shisha O (ed) Inequalities-III: proceedings of the third symposium on inequalities. Academic Press, New York, pp 159–175

Koch T, Achterberg T, Andersen E, Bastert O, Berthold T, Bixby R, Danna E, Gamrath G, Gleixner A, Heinz S, Lodi A, Mittelmann H, Ralphs T, Salvagnin D, Steffy D, Wolter K (2011) MIPLIB 2010. Math Program Comput 3(2):103–163

Kojima M, Mizuno S, Yoshise A (1989) A primaldual interior point algorithm for linear programming. In: Megiddo N (ed) Progress in mathematical programming: interior-point algorithms and related methods. Springer, New York, pp 29–47

Kojima M, Megiddo N, Mizuno S (1993) A primal-dual infeasible-interior-point algorithm for linear programming. Math Program 61(1):263–280

Koopmans T (1949) Optimum utilization of the transportation system. Econometrica 17(Supplement):136–146

Kunnumkal S, Talluri K, Topaloglu H (2012) A randomized linear programming method for network revenue management with product-specific no-shows. Transport Sci 46(1):90–108

Lee E, Gallagher R, Patterson D (2003) A linear programming approach to discriminant analysis with a reserved-judgment region. INFORMS J Comput 15(1):23–41

Lustig IJ, Marsten RE, Shanno DF (1994) Interior point methods for linear programming: computational state of the art. ORSA J Comput 6(1):1–14

Mansini R, Ogryczak W, Speranza M (2007) Conditional value at risk and related linear programming models for portfolio optimization. Ann Oper Res 152(1):227–256

Megiddo N (1983) Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems. SIAM J Comput 12(4):759–776

Megiddo N (1989) Pathways to the optimal set in linear programming. In: Megiddo N (ed) Progress in mathematical programming: interior-point algorithms and related methods. Springer, New York, pp 131–158

Mehrotra S (1992) On the implementation of a primal-dual interior point method. SIAM J Optim 2(4):575–601

Nadarajah S, Margot F, Secomandi N (2015) Relaxations of approximate linear programs for the real option management of commodity storage. Manage Sci 61(12):3054–3076

Padberg M (1999) Linear optimization and extensions. Algorithms and combinatorics, vol 12. Springer-Verlag

Press W, Teukolsky S, Vetterling W, Flannery B (2007) Numerical recipes. Cambridge University Press, New York

Raymond V, Soumis F, Orban D (2010) A new version of the improved primal simplex for degenerate linear programs. Comput Oper Res 37(1):91–98

Reid J (1982) A sparsity-exploiting variant of the Bartels–Golub decomposition for linear programming bases. Math Program 24(1):55–69

Romeijn H, Ahuja R, Dempsey J, Kumar A (2006) A new linear programming approach to radiation therapy treatment planning problems. Oper Res 54(2):201–216

Rong A, Lahdelma R (2008) Fuzzy chance constrained linear programming model for optimizing the scrap charge in steel production. Eur J Oper Res 186(3):953–964

Schrijver A (1998) Theory of linear and integer programming. Wiley, New York

Shamos M, Hoey D (1976) Geometric intersection problems. In: Seventeenth annual IEEE symposium on foundations of computer science, pp 208–215

Spielman D, Teng S (2004) Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. J ACM 51(3):385–463

Spitter J, Hurkens C, de Kok A, Lenstra J, Negenman E (2005) Linear programming models with planned lead times for supply chain operations planning. Eur J Oper Res 163(3):706–720

Strassen V (1969) Gaussian elimination is not optimal. Numer Math 13(4):354–356

Suhl L, Suhl U (1993) A fast LU update for linear programming. Ann Oper Res 43(1):33–47

Suhl U, Suhl L (1990) Computing sparse LU factorizations for large-scale linear programming bases. INFORMS J Comput 2(4):325–335

Tang L, Liu J, Rong A, Yang Z (2000) A mathematical programming model for scheduling steelmaking-continuous casting production. Eur J Oper Res 120(2):423–435

Terlaky T, Zhang S (1993) Pivot rules for linear programming: a survey on recent theoretical developments. Ann Oper Res 46(1):203–233

Todd M (1985) Linear and quadratic programming in oriented matroids. J Comb Theory 39(2):105–133

Tolla P (1986) A stable and sparsity exploiting LU factorization of the basis matrix in linear programming. Eur J Oper Res 24(2):247–251

Williams V (2012) An overview of the recent progress on matrix multiplication. ACM SIGACT News 34(3):57–69

Winston W (2004) Operations research: applications and algorithms. Duxbury Press, Belmont

Ye Y (2011) The simplex and policy-iteration methods are strongly polynomial for the markov decision problem with a fixed discount rate. Math Oper Res 36(4):593–603

Zhou P, Ang B (2008) Linear programming models for measuring economy-wide energy efficiency performance. Energy Policy 36(8):2911–2916