



OPEN ACCESS


Operations Research and Decisions

www.ord.pwr.edu.pl

OPERATIONS
RESEARCH
AND DECISIONS
QUARTERLY



A steepest feasible direction method for linear programming. Derivation and embedding in the simplex method

Biressaw C. Wolde¹ Torbjörn Larsson^{1*} 

¹Department of Mathematics, Linköping University, SE-58183 Linköping, Sweden

*Corresponding author; email address: torbjorn.larsson@liu.se

Abstract

A feasible direction method for linear programming has been proposed. The method is embedded in the framework of the simplex method, even though it works with non-edge feasible directions. The direction used is the steepest in the space of all variables or an approximation thereof, and it is found by solving a strictly convex quadratic program in the space of the nonbasic variables. Further, this program guarantees the feasibility of the direction even in the case of degeneracy. To remain within the simplex framework, the direction is represented by an auxiliary, or external, nonbasic column, which is a nonnegative linear combination of original nonbasic columns. We have made an experimental evaluation of the suggested method on both nondegenerate and highly degenerate problem instances. The overall results are very promising for continued research along this line, especially concerning various computational strategies that can be applied when the method is implemented.

Keywords: *linear programming, simplex method, steepest edge, feasible direction, external pivoting*

1. Introduction

Based on the type of paths they follow to reach optimality, the two fundamentally different and competitive approaches for solving linear programs are the simplex method and interior methods. The simplex method follows the edges of the feasible polyhedron while interior methods move through the relative interior of the feasible polyhedron. Interior methods may be classified into two broad categories based on their connection with the simplex method [17]: methods that work within the simplex framework and those that do not. Our linear programming approach can be categorized as a simplex-type interior method. It uses the steepest feasible directions, which are typically not along edges but can still be incorporated into the simplex framework through the introduction of auxiliary variables.

The simplex method for solving general linear programming problems was developed in the late 1940s. It has since then been an efficient tool for solving linear programs arising in various applications, and therefore immensely important within operations research. Borgwardt and Huhn [5] showed that the

average number of iterations needed by the standard version of the simplex method is polynomial, which explains its practical efficiency. However, by considering specially formulated problem instances, Klee and Minty [14] (see also, e.g., [18]) demonstrated that its worst-case performance is exponential. Much research has been devoted to various ways of enhancing the computational efficiency and numerical stability of the simplex method; an example of this is the design of efficient computational schemes for updating the basis inverse in each iteration. A crucial step in the simplex method is the pricing, or column selection. This step examines the reduced costs of the nonbasic variables that violate the optimality condition and then it selects one of them to enter the basis; the standard pricing rule is to simply find the most negative reduced cost (for a minimization problem).

More advanced rules for choosing the entering variables can significantly reduce the number of iterations needed to reach an optimal solution. Such rules are more elaborate and computationally demanding than the standard rule. However, because they lead to fewer simplex iterations they can still be very favourable for the practical efficiency of the overall simplex method. Several advanced pricing strategies have been suggested and implemented (see, e.g., [16]). Two examples of this are the steepest-edge rule and the Devex rule, which both decrease the number of iterations and the running time, as compared to using the standard pricing rule [9, 11, 13]. A good account of pricing rules in connection with the theoretical convergence properties of the simplex method can be found in [10].

Since optimality for any linear program is attained at an extreme point of the feasible polyhedron, it is sufficient to examine only such points to find an optimum; this is the foundation for the simplex method. In each nondegenerate iteration of the simplex method, it makes a move from a current extreme point along an edge of the polyhedron to an adjacent extreme point with a better objective value. Because moves are made along edges, the iterations become algebraically simple and computationally inexpensive.

The rationale for developing non-edge following simplex-type methods for linear programming is the convexity property of linear programs. Because of the convexity, it is always possible to move from any given, non-optimal feasible point along a feasible descent direction to an optimal solution. (The calculation of such a direction is however of course equivalent to solving the linear program.) Furthermore, given a basic feasible solution, such a direction can always be expressed as a nonnegative linear combination of the edge directions corresponding to the nonbasic variables.

Non-edge following methods that work within the simplex framework attempt to reduce the overall iterations and running times needed by the simplex method at the expense of additional computations in every iteration, or only some of them. The efficiency of using non-edge directions can however be expected to diminish as an optimal solution is approached, since at this stage the higher quality of a non-edge direction, as compared to the quality of the edge directions, do not warrant the additional computations that are needed. Further, non-edge directions typically lead to non-extreme points, and they are therefore clearly not effective in the ultimate search for an optimal extreme point. By embedding a non-edge following method in the simplex framework and allowing the overall method to fall back on a pure simplex strategy, these drawbacks are however easily dealt with.

One of the non-edge methods that tries to exploit these observations is the external pivoting principle of Eiselt and Sandblom [6, 7], which introduces auxiliary columns and corresponding variables into the linear program; these are referred to as external columns and variables. An external column is generated

as a positive linear combination of a subset of the current nonbasic columns, and an edge direction for an external variable then corresponds to a non-edge feasible direction in the original feasible polyhedron. The augmented linear program is solved by the standard simplex method, and letting an external variable enter the basis then corresponds to following a non-edge feasible direction in the original variable space. Several heuristic strategies for constructing external columns have been studied, and as reported by Eiselt and Sandblom [7], most of them decrease the number of simplex iterations quite considerably.

Another non-edge method is the feasible direction method of Murty and Fathi [8, 19], which has two phases: construction of a profitable direction and a reduction process. The profitable direction is formed as a nonnegative linear combination of the updated columns of the nonbasic variables with negative reduced costs (for a minimization problem), or a subset thereof. Further, the direction is represented by an auxiliary column. Two choices of weights are considered: equal weights and reduced cost weights. The reduction process is similar to the procedure of the ordinary simplex method. The method begins with a basic feasible solution and takes the maximal possible step in a profitable direction. The feasible solution reached is mostly not basic, and the reduction process then converts such a solution into a basic feasible solution without worsening the objective value. Convergence of the overall method is guaranteed under a nondegeneracy assumption. The direction-finding step and the auxiliary variable used by Murty and Fathi are clearly in essence equivalent to the techniques used by Eiselt and Sandblom [6, 7]. Mitra et al. [17] also constructed profitable directions by taking a linear combination of the eligible updated nonbasic columns using the negative of the reduced costs as weights.

In the same vein, Gondzio [12] has presented a practical non-edge feasible direction method for large sparse linear programming problems, which can be interpreted as an active set method. At a particular iteration, the method moves from a given feasible point along an improving feasible direction to a better feasible point. To find the feasible direction, the gradient of the objective function is projected approximately on the face defined by the constraints that are active at the given point. This step involves computing the inverse of a working basis, which is the submatrix of the active constraints.

More recently, Arsham [1] proposed still another, and rather opaque, non-edge feasible direction linear programming method that can be embedded in the simplex method. The method has three phases: an initialisation phase, a push phase, and a final iteration phase. The first two phases create an initial basic feasible solution, while the last phase searches for an optimal basic feasible solution by moving along non-edge directions that are obtained from projections of the full gradient of the objective function onto binding constraints. To embed this move in the simplex framework, the direction is replaced by an auxiliary variable, which is forced to enter the basis.

A still another non-edge linear programming method is the primal-dual algorithm (see, e.g., [18]). This is a purely dual method that works with dual feasible solutions that do not need to be extreme points. The non-edge direction used is the dual feasible direction that is steepest in the infinity norm. Since this norm can be formulated with linear constraints, the direction-finding problem becomes a linear program. The primal-dual algorithm does not rely on any of the mechanisms of the simplex method; hence it is not a simplex-type interior method but a genuine non-edge feasible direction method. It has gained the most attention for certain network optimization problems since the direction-finding linear program can then be efficiently solved by exploiting the network structure.

We present a feasible direction approach for general linear programming that can be incorporated into the simplex method through the use of external columns. The aim is to reduce the number of iterations and the running time of the simplex method. The generation of the feasible direction is founded on the steepest edge entering variable criterion [11] but the direction found is typically steeper than the steepest edge. Our approach differs from those in [6–8, 17, 19], and [12] in the way the feasible direction is constructed; in these works, *ad hoc* procedures are used, whereas we solve a convex quadratic programming problem. Further, this quadratic program includes constraints that guarantee that the direction becomes feasible even in the case of degeneracy.

The remainder of this paper is organised as follows. Section 2 introduces the problem, notations and some basic facts, while Section 3 gives the derivation of our steepest feasible direction method and describes how it can be embedded in the framework of the simplex method. Section 4 gives numerical results for the new method applied to randomly generated linear programs and linear programming relaxations of some medium-scale set covering problems; the former are nondegenerate, while the latter are highly degenerate. Finally, in Section 5 we draw some conclusions from our findings.

The work presented here is a continuation of that presented in the short conference publication [20]. The contributions made here are that we include a complete way of handling degeneracy, provide a review of related approaches, present some proofs (which was not done in [20]), and provide much more extensive computational results, both on nondegenerate and degenerate problem instances.

2. Preliminaries

Given a matrix $A \in \mathbb{R}^{m \times n}$ with $n > m$ and full rank, and the vectors $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$, we consider the linear program (LP)

$$\begin{aligned} xz^* &= \min z = c^T x \\ &\text{s.t. } Ax = b \\ &\quad x \geq 0 \end{aligned}$$

The feasible set of the problem is assumed to be non-empty and a non-singleton.

It is further assumed that a basic feasible solution is available. The index sets of the corresponding basic and nonbasic variables are denoted by \mathcal{B} and \mathcal{N} , respectively. Without loss of generality it can be assumed that $\mathcal{B} = \{1, \dots, m\}$ and $\mathcal{N} = \{m+1, \dots, n\}$. The basic feasible solution then corresponds to the partitioning $x = (x_B^T, x_N^T)^T$, with x_B and x_N being the vectors of basic and nonbasic variables, respectively. We also introduce the partitionings $A = (B, N)$ and $c = (c_B^T, c_N^T)^T$, where $B \in \mathbb{R}^{m \times m}$ is the non-singular basis matrix, $N \in \mathbb{R}^{m \times (n-m)}$ is the matrix of nonbasic columns, and c_B and c_N are the vectors of objective coefficients of the basic and nonbasic variables, respectively. The dual solution that is complementary to the primal basic feasible solution is given by $u^T = c_B^T B^{-1}$, and the vector of reduced costs of the nonbasic variables is $\bar{c}_N^T = c_N^T - u^T N$. Denoting the identity matrix of size m by I_m , problem LP can be equivalently expressed in the given basis as

$$\begin{aligned} z^* = \min z &= c_B^T B^{-1}b + \bar{c}_N^T x_N \\ \text{s.t. } I_m x_B + B^{-1}N x_N &= B^{-1}b \\ x_B, x_N &\geq 0 \end{aligned} \tag{1}$$

The basic feasible solution is $x_B = B^{-1}b \geq 0$ and $x_N = 0$, and it corresponds geometrically to an extreme point of the feasible polyhedron of the problem. If the basic solution is nondegenerate, then it is optimal if and only if $\bar{c}_N \geq 0$ holds. In the case of degeneracy, $\bar{c}_N \geq 0$ implies optimality, but optimality may hold even though $\bar{c}_N \not\geq 0$.

To introduce our approach, we first consider the nondegenerate case. Then a variable $x_j, j \in \mathcal{N}$, that enters the basis corresponds geometrically to a movement from the current extreme point in a feasible direction along an adjacent edge. Letting a_j be column $j \in \mathcal{N}$ of the matrix A and $e_{j-m} \in \mathbb{R}^{n-m}$ the unit vector with a one entry in position $j-m$, the edge direction is given by (see, e.g., [18])

$$\eta_j = \begin{pmatrix} -B^{-1}a_j \\ e_{j-m} \end{pmatrix} \in \mathbb{R}^n$$

Further, the columns of the matrix of edge directions,

$$(\eta_{m+1}, \eta_{m+2}, \dots, \eta_n) = \begin{pmatrix} -B^{-1}N \\ I_{n-m} \end{pmatrix}$$

where I_{n-m} is the identity matrix of size $n-m$ is a basis for the null space of A , and under nondegeneracy they span the cone of feasible directions from the current extreme point. (In the case of degeneracy, directions in this cone may be infeasible.)

With $\|\cdot\|$ denoting the Euclidean norm, the directional derivative of the objective function along a normalised edge direction is

$$\frac{c^T \eta_j}{\|\eta_j\|} = \frac{(c_B^T, c_N^T) \eta_j}{\|\eta_j\|} = \frac{(-c_B^T B^{-1}a_j + c_j)}{\|\eta_j\|} = \frac{\bar{c}_j}{\|\eta_j\|}$$

This directional derivative is used in the steepest-edge criterion [11] for choosing entering variable in the simplex method. It finds a variable $x_r, r \in \mathcal{N}$, to enter the basis such that $r \in \arg \min_{j \in \mathcal{N}} \bar{c}_j / \|\eta_j\|$.

We next consider arbitrary feasible directions, constructed as nonnegative linear combinations of the edge directions. Denoting the weights in the combination by $w \in \mathbb{R}_+^{n-m} \setminus \{0\}$, these directions are

$$\eta(w) = \begin{pmatrix} \eta_B(w) \\ \eta_N(w) \end{pmatrix} = \sum_{j \in \mathcal{N}} w_j \eta_j = \begin{pmatrix} -B^{-1}N \\ I_{n-m} \end{pmatrix} w = \begin{pmatrix} -B^{-1}Nw \\ w \end{pmatrix}$$

with $\eta_B(w)$ and $\eta_N(w)$ being the components in the basic and nonbasic spaces, respectively. Notice that any feasible solution can be reached from the given extreme point along some such direction, and that this in particular holds for an optimal solution. (This property holds also under degeneracy.) Notice also

that $\eta_B(w) = -B^{-1}(a_{m+1}, \dots, a_n)w = -B^{-1} \sum_{j \in \mathcal{N}} w_j a_j$, that is, the negative of a nonnegative linear combination of the original columns $(a_j)_{j \in \mathcal{N}}$ expressed in the current basis, and that the directional derivative

$$c^T \eta(w) = (c_B^T, c_N^T) \begin{pmatrix} -B^{-1}N \\ I_{n-m} \end{pmatrix} w = (c_N^T - c_B^T B^{-1}N) w = \bar{c}_N^T w$$

3. Derivation

The foundation for our development is the problem and result below. Letting $\text{supp}(\cdot)$ denote the support of a vector, that is, its number of non-zero components, we define the steepest-edge problem (SEP)

$$\begin{aligned} \min \quad & \bar{c}_N^T w \\ \text{s.t.} \quad & \|\eta(w)\|^2 \leq 1 \\ & \text{supp}(w) = 1 \\ & w \geq 0 \end{aligned}$$

Proposition 1. An index $r \in \mathcal{N}$ fulfils the steepest-edge criterion

$$r \in \arg \min_{j \in \mathcal{N}} \frac{\bar{c}_j}{\|\eta_j\|}$$

if and only if the solution

$$w_j = \begin{cases} 1/\|\eta_j\| & \text{if } j = r \\ 0 & \text{otherwise} \end{cases}, \quad j \in \mathcal{N}$$

solves SEP.

Proof. The problem SEP has $|\mathcal{N}|$ feasible solutions, say w^k , $k \in \mathcal{N}$, given by

$$w_l^k = \begin{cases} 1/\|\eta_l\| & \text{if } l = k \\ 0 & \text{otherwise} \end{cases}, \quad l \in \mathcal{N}$$

Further, $\bar{c}_N^T w^k = \sum_{l \in \mathcal{N}} \bar{c}_l w_l^k = \bar{c}_k / \|\eta_k\|$. The result follows. \square

Notice that the optimal value of SEP coincides with the steepest-edge slope $\bar{c}_r / \|\eta_r\|$.

By relaxing the support constraint in problem SEP, it may be possible to find a feasible direction that is steeper than the steepest-edge direction. The relaxed problem is referred to as the direction-finding problem (DFP) and can be expressed as

$$\begin{aligned} \min \quad & \bar{c}_N^T w \\ \text{s.t.} \quad & w^T Q w \leq 1 \\ & w \geq 0 \end{aligned} \tag{2}$$

where the matrix

$$Q = \begin{pmatrix} -B^{-1}N \\ I_{n-m} \end{pmatrix}^T \begin{pmatrix} -B^{-1}N \\ I_{n-m} \end{pmatrix} = N^T B^{-T} B^{-1} N + I_{n-m} \in \mathbb{R}^{(n-m) \times (n-m)}$$

is symmetric and positive definite. If $\bar{c}_N \geq 0$ holds, then the zero solution is optimal in problem DFP, and otherwise, it has a unique, nonzero optimal solution with a negative objective value, and that fulfils the normalisation constraint (2) with equality. Such a non-zero optimal solution defines the steepest feasible direction of descent for problem LP from the given extreme point; in general this direction is a non-trivial nonnegative linear combination of the edge directions and has a directional derivative that is strictly better than that of the steepest-edge direction.

Example 3.1. Consider the linear program

$$\begin{aligned} z^* = \min z &= -x_1 - 2x_2 \\ \text{s.t. } 5x_1 - 2x_2 &\leq 10 \\ -2x_1 + 4x_2 &\leq 8 \\ 2x_1 + x_2 &\leq 6 \\ x_1, x_2 &\geq 0 \end{aligned}$$

which has optimal solution $(1.6, 2.8)^T$.

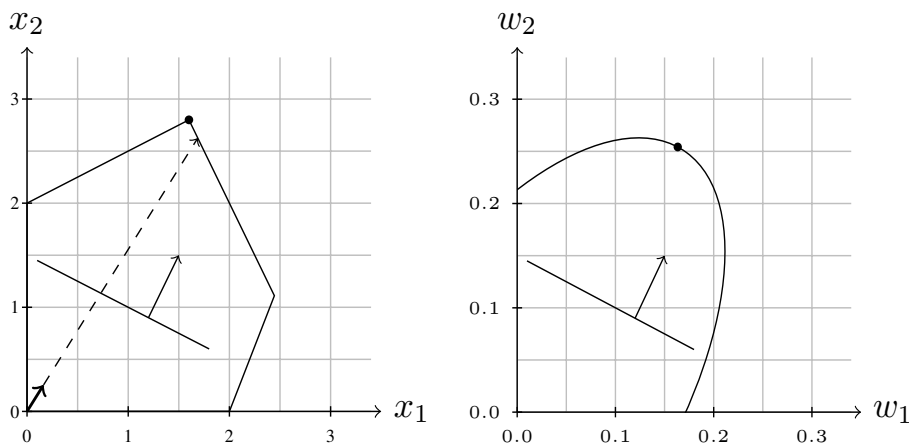


Figure 1. Illustration of steepest feasible direction and problem DFP

The problem is depicted to the left in Figure 1. We consider the extreme point at the origin, which has the slack basis $B = I_3$. Then $\eta_1 = (-5, 2, -2, 1, 0)^T$ and $\eta_2 = (2, -4, -1, 0, 1)^T$, with $\bar{c}_1/\|\eta_1\| = -1/\sqrt{34}$ and $\bar{c}_2/\|\eta_2\| = -2/\sqrt{22}$. Hence, with the steepest-edge criterion, the variable x_2 would enter the basis. The picture to the right in Figure 1 shows the feasible set of DFP. Its optimal solution is $w^* \approx (0.163, 0.254)^T$ (marked by the dot and shown as the short thick vector in the figure to the left) with $\bar{c}_N^T w^* = c^T \eta(w^*) \approx -0.672$, which shall be compared with $-2/\sqrt{22} \approx -0.426$. The feasible direction found is $\eta(w^*) \approx (-0.309, -0.690, -0.581, 0.163, 0.254)^T$. The maximal feasible step in this direction is about 10.3 (shown as the dashed vector) and yields the boundary point $(x_1, x_2) \approx (1.687, 2.625)$, which

has an objective value that is better than those of the two extreme points that are adjacent to the origin. Further, this point is close to the optimal solution.

Notice that in an optimal solution w^* to DFP, $w_j^* > 0$ can hold even when $\bar{c}_j > 0$, which means that the steepest feasible direction can include edge directions that alone give ascent. An example of this is obtained if the objective function above is changed to $\min z = 2x_1 - 5x_2$, which results in optimal weights $w^* \approx (0.065, 0.250)^T$.

Under degeneracy, a steepest-edge direction can be infeasible, and similarly, the steepest direction obtained from DFP can be infeasible. Further, the current basic feasible solution can be optimal even though a steepest-edge direction or a steepest direction from DFP indicates non-optimality. We next describe how to properly handle degeneracy.

Let $\bar{b} = B^{-1}b$ and denote the i th row in $B^{-1}N$ by \bar{a}_i^T . Then the constraint (1) can be written as

$$x_i + \bar{a}_i^T x_N = \bar{b}_i, \quad i \in \mathcal{B}$$

Let $\mathcal{B}^0 = \{i \in \mathcal{B} \mid \bar{b}_i = 0\}$ be the index set of the degenerate basic variables. The following easily proven result states how to effectively tackle degeneracy by using explicit degeneracy-breaking constraints.

Proposition 2. Let $w \in \mathbb{R}_+^{n-m} \setminus \{0\}$. Then the direction $\eta(w)$ is feasible in problem LP from the given basic feasible solution if and only if $\bar{a}_i^T w \leq 0$ holds for all $i \in \mathcal{B}^0$.

Hence, to handle degeneracy the following degenerate DFP (DDFP) should be used.

$$\begin{aligned} \min \quad & \bar{c}_N^T w \\ \text{s.t.} \quad & w^T Q w \leq 1 \\ & \bar{a}_i^T w \leq 0, \quad i \in \mathcal{B}^0 \\ & w \geq 0 \end{aligned} \tag{3}$$

Further, it can correctly verify optimality, as stated below.

Theorem 1. The basic feasible solution $x_B = B^{-1}b$ and $x_N = 0$ is optimal in problem LP if and only if the zero solution is optimal in DDFP.

Proof. The basic feasible solution is optimal if and only if there is no feasible descent direction. Using that $\bar{c}_N^T w = c^T \eta(w)$, this holds exactly when the zero solution is optimal in DDFP. \square

If the basic feasible solution is not optimal, then the non-zero optimal solution to DDFP is unique, has a negative objective value, fulfils the constraint (3) with equality, and provides the steepest feasible direction of descent for problem LP. As for DFP, this direction is in general a non-trivial combination of edge directions.

The constraint (3) in DDFP is bounding the length of the direction found and thereby ensures that the optimal value is finite. It will next be established that problem DDFP can however be solved by using a problem where this constraint has been relaxed. Let $\mu/2 > 0$ be a Lagrangian multiplier for constraint (3) and consider the Lagrangian Relaxed DDFP (RDDFP)

$$\begin{aligned} \min r(w) &= \bar{c}_N^T w + \frac{\mu}{2} w^T Q w \\ \text{s.t. } \bar{a}_i^T w &\leq 0, \quad i \in \mathcal{B}^0 \\ w &\geq 0 \end{aligned}$$

The constant term $-\mu/2$ is of no interest and therefore omitted. The objective function r is strictly convex and therefore problem RDDFP has a unique optimum, which is denoted $w^*(\mu)$. As stated below, this optimum can easily be scaled into an optimal solution to DDFP.

Theorem 2. If $w^*(\mu) = 0$, then the zero solution is optimal in problem DDFP, and otherwise the solution $w^* = w^*(\mu)/\|\eta(w^*(\mu))\|$ is optimal in DDFP.

Proof. To prove the statement for $w^*(\mu) = 0$ we use that problems DDFP and RDDFP are both convex and have relative interior points. Hence, if $w^*(\mu) = 0$ solves RDDFP then it is a Karush–Kuhn–Tucker point for this problem, and since $\nabla r(0) = \bar{c}_N$ and $w^*(\mu)^T Q w^*(\mu) = 0 < 1$ hold, it is then also a Karush–Kuhn–Tucker point for problem DDFP, and therefore solves this problem.

To prove the statement for the case $w^*(\mu) \neq 0$ we use Everett's theorem (see, e.g., [2]). From this theorem follows that if $w^*(\mu)$ solves RDDFP, then it also solves the problem

$$\begin{aligned} \min \bar{c}_N^T w \\ \text{s.t. } w^T Q w &\leq w^*(\mu)^T Q w^*(\mu) \\ \bar{a}_i^T w &\leq 0, \quad i \in \mathcal{B}^0 \\ w &\geq 0 \end{aligned}$$

Using that $w^*(\mu)^T Q w^*(\mu) = \|\eta(w^*(\mu))\|^2$ and introducing the scaled variable vector $v = w/\|\eta(w^*(\mu))\|$, this problem is equivalent to

$$\begin{aligned} \min \|\eta(w^*(\mu))\| \bar{c}_N^T v \\ \text{s.t. } v^T Q v &\leq 1 \\ \bar{a}_i^T v &\leq 0, \quad i \in \mathcal{B}^0 \\ v &\geq 0 \end{aligned}$$

which is in turn equivalent to problem DDFP, which proves the statement. \square

Hence, the quadratic program RDDFP will, for any choice of $\mu/2 > 0$ provide the steepest feasible direction of descent for LP.

Our next result gives an interesting characterisation of the gradient of the objective function r . It could be useful for streamlining computations if problem RDDFP is approached by an iterative descent method (see, e.g., [4]). The reader may recall that $\eta_B(w) = -B^{-1}Nw$.

Proposition 3. Let $\Delta u^T = \mu \eta_B(w)^T B^{-1}$. Then $\nabla r(w) = c_N - N^T (u + \Delta u) + \mu w$.

Proof. Inserting $\Delta u = -\mu B^{-T} B^{-1} N w$ gives that

$$\nabla r(w) = c_N - N^T (u + \Delta u) + \mu w = c_N - N^T u + \mu (N^T B^{-T} B^{-1} N w + I_{n-m}) w = \bar{c}_N + \mu Q w$$

□

Notice that the expression for Δu is similar to that of a complementary dual solution. Further, the expression for the gradient $\nabla r(w)$ reveals that it can be computed by using the pricing mechanism of the simplex method but with a modified dual solution.

We next consider how to utilise a nonzero optimal solution to DDFP; it is also possible to utilise an approximate solution to DDFP provided that it is feasible and $c^T \eta(w^*) = \bar{c}_N^T w^* < 0$ holds. To make use of the non-edge feasible descent direction $\eta(w^*)$ within the framework of the simplex method, it is translated into an external column [6, 7], which is simply a nonnegative linear combination of the original columns in LP. The objective coefficient of the external column, which is indexed by $n + 1$, is $c_{n+1} = c_N^T w^*$ and the vector of constraint coefficients is $a_{n+1} = N w^*$. Problem LP is then augmented with the external column and a corresponding external variable, x_{n+1} , giving

$$\begin{aligned} \min z &= c^T x + c_{n+1} x_{n+1} \\ \text{s.t. } Ax + a_{n+1} x_{n+1} &= b \\ x, x_{n+1} &\geq 0. \end{aligned}$$

This augmented problem LP is, in essence, equivalent to the original problem LP, as stated below; this result is easily verified.

Proposition 4. If an external column is constructed at the basic feasible solution $(x_B, x_N) = (B^{-1}b, 0)$ and the solution (\bar{x}, \bar{x}_{n+1}) with $\bar{x}_{n+1} > 0$ is feasible in the augmented LP, then the solution

$$\hat{x} = (\hat{x}_B, \hat{x}_N) = (B^{-1}b - B^{-1}N(\bar{x}_N + \bar{x}_{n+1}w^*), \bar{x}_N + \bar{x}_{n+1}w^*)$$

is feasible in LP with $c^T \hat{x} = c^T \bar{x} + c_{n+1} \bar{x}_{n+1}$.

Although the addition of the external variable does not alter the given problem in any essential way, it introduces a linear dependency between the columns in the problem (which should however not be difficult to handle for a modern solver) and it can also introduce alternative optimal solutions.

The result of Proposition 4 can be used for recovering an optimal solution to the original LP whenever the external variable takes a positive value in the optimal solution found. The result also has implications concerning the possibility of an external column to remain basic in an optimal solution. If some variable x_j , $j \in \mathcal{N}$, with $w_j^* > 0$, is nonbasic in all optimal solutions to LP, then the external variable must be nonbasic in all optimal solutions to the augmented LP. Conversely, if an optimal value of x_{n+1} is strictly positive, then there is an optimal solution to LP where all variables x_j , $j \in \mathcal{N}$, with $w_j^* > 0$ are basic. Hence, the external column can, loosely speaking, remain in the basis at optimality only if all columns that it includes are basic at optimality.

Letting $\bar{c}_{n+1} = c_{n+1} - u^T a_{n+1}$, the augmented LP expressed in the current basis is

$$\begin{aligned} z^* = \min z &= c_B^T B^{-1} b + \bar{c}_N^T x_N + \bar{c}_{n+1} x_{n+1} \\ \text{s.t. } I_m x_B + B^{-1} N x_N + B^{-1} a_{n+1} x_{n+1} &= B^{-1} b \\ x_B, x_N, x_{n+1} &\geq 0 \end{aligned}$$

Notice that $B^{-1} a_{n+1} = B^{-1} N w^* = -\eta_B(w^*)$ and that $\bar{c}_{n+1} = c_N^T w^* - u^T N w^* = \bar{c}_N^T w^* < 0$. By letting the external column enter the basis, the feasible direction will be followed. If $B^{-1} a_{n+1} \leq 0$ holds, then the optimal objective value of LP is unbounded. Otherwise, the step taken in the direction is given by the value that the external variable obtains.

4. Numerical experiments

We present the results of some numerical experiments with the use of external columns constructed from steep feasible directions. The aim of the experiments is only to make a preliminary assessment of the potential benefit of using this principle within the simplex method. We use a straightforward MATLAB implementation of the revised simplex method with the standard Dantzig entering variable criterion but allow ordinary pivots to be replaced by pivots on external columns. To handle degeneracy, Bland's second rule (see, e.g., [18]), which selects a leaving variable based on the smallest index is used.

We have performed experiments on randomly generated problem instances and linear programming relaxations of real-life set covering problem instances; the former are nondegenerate while the latter are highly degenerate. The randomly generated instances are created according to the principle used in [7]; they are of maximization type. The set covering instances is taken from OR-Library [3]. Sizes of the instances used are included in Tables 1–3.

The simplex method is initialised at a basic feasible solution. The nondegenerate problems are inequality-constrained and the origin is feasible, and hence we initialise with the slack basis. For the degenerate problems we use an *ad hoc* heuristic method to construct an integer-valued basic feasible solution; the resulting basis is typically highly degenerate (for the instances used).

For large instances of problem LP the quadratic program RDDFP can be too demanding to solve, considering its sole purpose of giving a feasible direction, even though this direction is expected to be of high quality. This justifies the use of a restricted version of problem RDDFP, which includes only a subset $J \subset \mathcal{N}$ of the edge directions. Let $w_J = (w_j)_{j \in J}$, $\bar{c}_J = (\bar{c}_j)_{j \in J}$, $N_J = (a_j)_{j \in J}$, $Q_J = N_J^T B^{-T} B^{-1} N_J + I_{|J|}$, and let \bar{a}_{iJ}^T be the i th row in $B^{-1} N_J$. The restricted RDDFP then is

$$\begin{aligned} \min r_J(w_J) &= \bar{c}_J^T w_J + \frac{\mu}{2} w_J^T Q_J w_J \\ \text{s.t. } \bar{a}_{iJ}^T w_J &\leq 0, \quad i \in \mathcal{B}^0 \\ w_J &\geq 0 \end{aligned}$$

Notice that even though RDDFP always finds a feasible descent direction, the restricted problem may fail to do so, since the edge directions included therein do not span all feasible directions. If the restricted problem is feasible, then it will give an approximate steepest feasible descent direction. A small

value of $|J|$ makes the problem computationally cheap but at the expense of the quality of the feasible direction found. A high value of $|J|$ makes the problem more expensive, but it can then also yield a steeper direction which results in fewer simplex iterations; this trade-off is studied in the experiments. In our implementation, the restricted RDDFP is solved using the built-in MATLAB solver quadprog.

Even though an optimal solution to RDDFP can include edge directions with positive reduced costs, it is still reasonable to construct a restriction by only selecting edges with negative reduced costs. Two ways of constructing the set J are considered, referred to as *Dantzig selection* and *steepest-edge selection*, respectively. With k denoting the number of edge directions included in the restricted RDDFP, these are to find the k most negative values among $\{\bar{c}_j\}_{j \in \mathcal{N}}$ and $\{\bar{c}_j/\|\eta_j\|\}_{j \in \mathcal{N}}$, respectively; the latter selection is of course computationally more costly. The constructed restrictions of RDDFP contain only edge directions that correspond to original columns, although it is in principle possible to use an already generated external columns to define a new external column.

The restrictions of RDDFP used include small portions of the edge directions with negative reduced costs. For the nondegenerate instances, the restricted RDDFP includes the 5%, 10% or 20% of these directions that are best according to the criterion used (Dantzig or steepest-edge). For the degenerate instances, which contain much more variables, the best 0.5%, 1% or 10% are included.

For the degenerate instances, we let the set J contain also several additional edge directions, which are included to span the set of feasible directions sufficiently well to allow the restricted RDDFP to find a feasible direction. These additional edge directions are randomly chosen (without any regard to their reduced costs). It is reasonable to let their number be related to the number of degenerate basic variables, and we tried using 5, 10 or 20 times the number of degenerate basic variables.

From the derivation of the problem DFP follows that it can provide non-edge feasible directions that are steeper than the steepest-edge direction. Our first experiment illustrates that the restricted RDDFP can also have this feature even when the simple Dantzig selection is used to construct a moderately sized set J . We also compare with two *ad hoc* rules for constructing feasible directions; the names of these rules, M2 and M3, are taken from [6, 7]. To state them, we let $\mathcal{N}^- = \{j \in \mathcal{N} \mid \bar{c}_j < 0\}$. In both rules, $w_j = 0$, $j \in \mathcal{N} \setminus \mathcal{N}^-$. The rule M2 is to use $w_j = -\bar{c}_j$, $j \in \mathcal{N}^-$, and the rule M3 is to use $w_j = -\bar{c}_j \min\{\bar{b}_i/\bar{a}_{ij} \mid \bar{a}_{ij} > 0, i \in \mathcal{B}\}$, $j \in \mathcal{N}^-$. Rule M3 is quite computationally demanding since it includes the minimum ratio criterion of the simplex method for all possible entering variables. (The rule M1 is to use $w_j = 1$, $j \in \mathcal{N}^-$. It is however in general inferior to M2, and therefore not considered.)

To compare the quality of various feasible directions, we normalise their lengths and calculate directional derivatives. We first give results for two nondegenerate instances, of sizes $1,000 \times 2,000$ and $500 \times 5,000$. The directional derivatives of the steepest-edge directions at the initial bases are for these instances -0.116 and -0.182 , respectively. For both these instances the rules M2 and M3 give almost identical feasible directions (after normalisations); this is because the random generation yields values of $\min\{\bar{b}_i/\bar{a}_{ij} \mid \bar{a}_{ij} > 0, i \in \mathcal{B}\}$ that are similar for all $j \in \mathcal{N}$. For the two instances, the rules M2/M3 give feasible directions with directional derivatives -0.165 and -0.230 , respectively; hence they are steeper than the steepest-edge directions. We compare them to directions obtained from restricted RDDFP problems that include the 5%, 10% or 20% best edges according to the Dantzig selection. For the two instances, the directional derivatives of the feasible directions found are in the ranges of -0.244 to -0.247

and -0.378 to -0.384 , respectively. Hence, the directions are even steeper than those found by the rules M2/M3. (Using more than 20% best edges gives only marginally better directions.)

We next consider the degenerate instance rail507. This problem contains 63,009 variables, out of which 17,539 have negative reduced costs in the initial basis. Out of these, only 90 (about 0.51%) yield a nondegenerate pivot. In particular, the entering variables obtained from the Dantzig or steepest-edge criteria yield degenerate pivots. Among the edges that yield a nondegenerate pivot, the steepest has a directional derivative of -0.707 . Rule M2 does not give a feasible direction; this is due to the non-zero weights on all the edge directions, among which most are infeasible and cause the weighted direction to become infeasible. The rule M3 however puts non-zero weights only on feasible edge directions (since $w_j = 0$ if $\bar{b}_i = 0$ for some $\bar{a}_{ij} > 0$), and it is therefore, by convexity, guaranteed to give a feasible direction. The directional derivative of the feasible direction found by M3 is -1.108 . We compare this with the directional derivative of the feasible direction produced by the restricted RDDFP when using the 0.5%, 1%, or 10% best columns according to the Dantzig selection, together with 5, 10 or 20 times the number of degenerate basic variables of additional, randomly chosen columns. These restricted RDDFP problems always produce feasible directions—even for the most restrictive choices—and their directional derivatives are in the range -1.173 to -2.050 , obtained from the smallest and largest restricted RDDFP, respectively. Hence, these directions are all steeper than the direction found by the rule M3.

Although a non-edge direction obtained from the restricted RDDFP is advantageous compared to edge directions, it is more computationally costly to find. Further, as mentioned in Section 1, non-edge directions are likely to be less efficient as an optimal solution is approached, concerning their computational cost, and typically they do not lead to extreme points in the feasible polyhedron. It is therefore reasonable to combine pivots on external columns with ordinary simplex pivots, and in particular it is reasonable to fall back to the latter in later iterations.

For the nondegenerate problem instances, the solution process alternates between the generation of external columns that directly enter the basis and the use of the standard simplex method. The process always starts with the generation of an external column, but for the continuation, we have tried three strategies: (i) no more external column is generated, (ii) external columns are generated with regular intervals (concerning simplex pivots), and (iii) an external column is generated when the latest external column leaves the basis. In the latter two strategies, the generation of external columns is terminated when the number of negative reduced costs is below a low threshold, which we set to 10, since this indicates that the number of remaining simplex iterations is small. For the degenerate problem instances, we only present results for the strategy of generating an external column when the latest external column exits the basis, since this turned out to be the best strategy.

We study the number of simplex iterations and the running times needed to reach optimality when using external columns based on approximate steepest feasible directions, and when using the various solution strategies, as compared to when using the standard simplex method. The simplex iterations and running times used for the nondegenerate problem instances are given in Tables 1 and 2. Figure 2 shows the convergence histories for the problem instance of size $1,000 \times 2,000$ when using a single external column that is calculated from 5%, 10% or 20% of the edge directions that are best according to the Dantzig or steepest-edge selections.

Table 1. Results for nondegenerate instances¹

Problem size $m \times n$	Parameters		Dantzig selection				Steepest-edge selection			
	port	max	iter	time	ext	size	iter	time	ext	size
1,000 × 2,000	–	–	35,181	418.0	–	–	–	–	–	–
	5	∞	17,881	183.5	1	100.0	20,503	214.4	1	100.0
	5	1,000	17,295	179.3	18	17.6	19,778	214.7	20	21.2
	5	exit	17,512	183.4	6	28.0	20,611	222.3	5	28.6
	10	∞	21,460	225.9	1	200.0	22,160	234.5	1	200.0
	10	1,000	19,072	200.5	20	30.1	20,336	223.6	21	30.8
	10	exit	19,685	207.1	9	35.7	17,757	186.9	2	111.0
	20	∞	20,244	217.6	1	400.0	21,110	226.7	1	400.0
	20	1,000	20,098	221.4	21	62.7	21,162	237.0	22	67.9
20	exit	19,380	201.7	5	111.8	21,159	229.3	2	231.5	
1,000 × 4,000	–	–	57,572	1,096.1	–	–	–	–	–	–
	5	∞	37,823	571.8	1	200.0	28,582	443.6	1	200.0
	5	1,000	36,749	558.5	37	25.6	28,450	446.9	29	25.7
	5	exit	38,334	552.7	5	49.2	28,188	427.9	2	109.0
	10	∞	32,322	486.4	1	400.0	32,598	489.9	1	400.0
	10	1,000	33,358	514.6	34	46.3	31,758	507.8	32	47.8
	10	exit	33,068	519.5	4	115.5	33,127	515.9	4	110.0
	20	∞	34,024	504.5	1	800.0	32,887	499.4	1	800.0
	20	1,000	33,861	527.8	34	95.1	31,218	500.1	32	103.3
	20	exit	34,003	507.9	9	127.0	32,371	494.1	4	232.3
2,000 × 3,000	–	–	104,123	4,606.6	–	–	–	–	–	–
	5	∞	64,174	2,338.5	1	150.0	56,275	2,044.7	1	150.0
	5	1,000	62,890	2,452.5	63	23.3	60,817	2,473.1	61	20.5
	5	exit	62,680	2,399.5	14	23.6	56,787	2,191.4	5	38.8
	10	∞	72,156	2,714.8	1	300.0	54,484	1,992.5	1	300.0
	10	1,000	62,764	2,413.7	63	42.2	56,348	2,234.2	57	42.6
	10	exit	71,116	2,784.3	13	43.9	54,876	2,117.9	6	61.3
	20	∞	60,708	2,222.1	1	600.0	61,965	2,247.0	1	600.0
	20	1,000	56,299	2,124.2	57	85.0	62,368	2,483.6	63	78.8
	20	exit	60,087	2,277.9	13	75.9	58,193	2,230.3	4	176.0
800 × 1,500	–	–	19,206	134.0	–	–	–	–	–	–
	5	∞	12,351	74.8	1	75.0	11,237	67.0	1	75.0
	5	1,000	12,842	81.7	13	15.6	11,490	72.1	12	16.3
	5	exit	12,214	75.2	7	19.4	9,649	58.2	2	43.5
	10	∞	10,895	63.8	1	150.0	10,722	63.2	1	150.0
	10	1,000	10,986	68.3	11	35.8	10,068	63.1	11	29.7
	10	exit	11,112	67.6	8	29.4	9,709	58.1	3	59.7
	20	∞	10,728	64.1	1	300.0	10,695	63.4	1	300.0
	20	1,000	12,879	83.2	13	55.7	10,636	68.3	11	53.2
	20	exit	12,003	75.1	10	43.1	10,471	66.2	2	166.0

¹ m and n are the numbers of constraints and variables, respectively, port is the portion of the edge directions with negative reduced costs that are included in the restricted RDDFP, max is the number of simplex iterations between the generation of external columns, iter and time are the number of simplex iterations used and the running time, respectively, ext is the number of external columns generated, and the size is the average number of edge directions included in the RDDFP problem. Further, ∞ means that an external column is generated at the initial basis only, and exit means that an external column is generated when the latest external column exits the basis. For each instance, the first line gives the number of simplex iterations and the running time with the standard simplex method. The three best results for each instance and selection strategy, concerning iterations and running time, respectively, are shown in boldface.

Table 2. Results for nondegenerate instances (notations as in Table 1)

Problem size $m \times n$	Parameters		Dantzig selection				Steepest-edge selection			
	port	max	iter	time	ext	size	iter	time	ext	size
$800 \times 3,000$	–	–	35,023	378.5	–	–	–	–	–	–
	5	∞	20,171	175.6	1	150.0	19,225	169.5	1	150.0
	5	1,000	20,269	181.7	21	23.5	20,588	188.1	21	23.9
	5	exit	19,684	171.5	3	56.7	19,176	169.5	5	38.0
	10	∞	20,967	192.7	1	300.0	19,129	172.1	1	300.0
	10	1,000	21,589	192.8	22	43.1	19,497	177.3	20	39.3
	10	exit	19,726	171.4	12	46.8	17,160	147.1	2	159.0
	20	∞	20,103	188.7	1	600.0	18,834	169.5	1	600.0
	20	1,000	18,763	165.6	19	92.7	18,293	168.7	19	88.1
20	exit	19,510	181.8	8	96.9	18,185	161.3	4	165.0	
$800 \times 4,500$	–	–	38,140	582.9	–	–	–	–	–	–
	5	∞	28,322	334.4	1	225.0	22,920	262.9	1	225.0
	5	1,000	23,569	268.1	24	35.5	21,731	259.6	22	32.4
	5	exit	26,331	302.1	6	46.5	22,462	257.1	3	83.0
	10	∞	22,130	253.8	1	450.0	21,007	245.8	1	450.0
	10	1,000	23,335	270.9	24	54.3	18,772	220.2	19	63.6
	10	exit	20,246	236.1	5	100.8	21,358	245.6	7	74.7
	20	∞	22,533	264.9	1	900.0	24,324	295.7	1	900.0
	20	1,000	23,272	270.8	24	127.8	20,184	242.3	21	102.2
20	exit	22,869	268.2	12	97.0	22,975	264.2	3	321.0	

For the nondegenerate problem instances, we conclude that it is possible to significantly reduce both the number of simplex iterations and the running time—sometimes up to around 50%—by using external columns derived from steep feasible directions. As can be expected, the steepest-edge selection criterion often gives a slightly superior performance with respect to iterations, but also with respect to running time even though it is computationally more demanding. The overall results are however surprisingly insensitive to the choice of selection criterion and parameter values; we believe that this supports the general principle of using external columns that are derived from steep feasible directions.

The results for the degenerate problem instances are given in Table 3. Figure 3 shows the convergence histories for the problem instance `scpnrg1`.

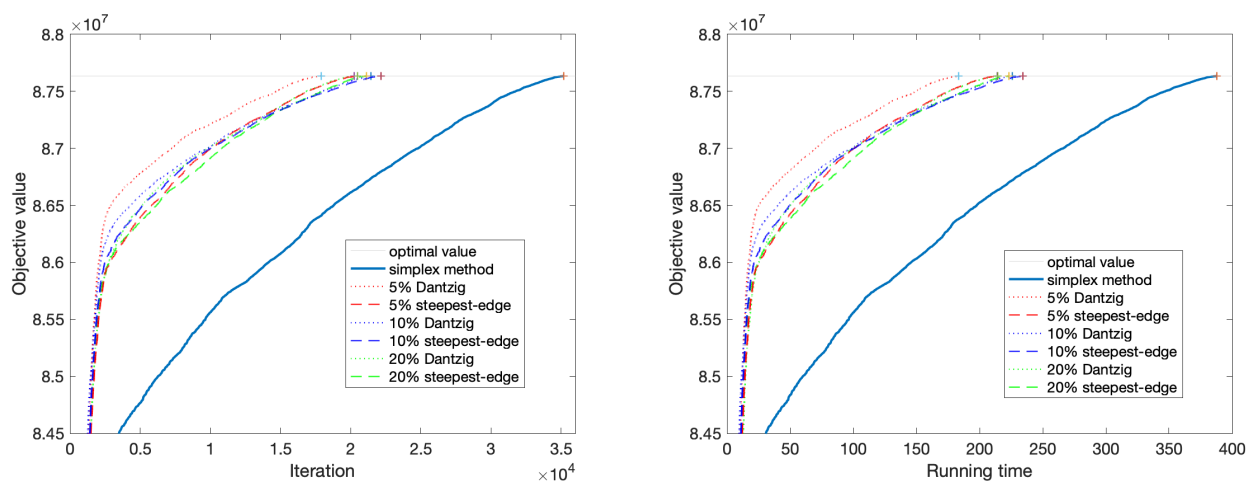


Figure 2. Objective value versus simplex iteration and running time, respectively, for the instance of size $1,000 \times 2,000$

Table 3. Results for degenerate instances¹

Problem size $m \times n$	Parameters		Dantzig selection				Steepest-edge selection			
	port	mult	iter	time	ext	size	iter	time	ext	size
507 × 63,009 [rail507]	–	–	23,002	182.4	–	–	–	–	–	–
	0.5	5	19,436	165.7	59	357.7	16,477	150.1	21	374.9
	0.5	10	16,013	147.1	42	666.5	18,736	181.7	19	878.3
	0.5	20	17,172	206.3	38	1,232.8	21,496	291.5	32	1,624.9
	1	5	19,260	157.7	45	392.1	17,226	157.3	16	418.4
	1	10	18,711	174.1	52	659.0	18,826	180.2	20	777.5
	1	20	18,223	242.9	45	1,420.3	17,767	208.3	17	1,655.2
	10	5	17,208	175.1	31	1,107.8	18,117	176.7	18	1,005.1
	10	10	16,144	166.8	29	1,265.5	15,377	151.4	13	1,426.2
10	20	16,262	224.1	25	2,008.4	17,778	290.6	25	2,191.5	
516 × 47,311 [rail516]	–	–	15,462	108.3	–	–	–	–	–	–
	0.5	5	11,184	69.3	18	654.8	10,387	72.1	11	700.7
	0.5	10	9,378	82.9	21	1,305.0	11,603	93.1	11	1,477.1
	0.5	20	9,948	160.3	17	2,569.3	9,738	134.6	11	2,764.4
	1	5	10,577	73.6	27	715.5	10,136	77.4	15	771.7
	1	10	9,910	101.7	27	1,398.2	11,148	101.1	15	1,467.3
	1	20	11,833	236.2	22	2,799.8	10,396	210.4	17	2,994.6
	10	5	11,726	95.9	16	1,489.7	9,805	86.5	10	1,668.5
	10	10	11,413	113.0	13	2,133.1	8,824	91.4	9	2,150.4
10	20	13,853	305.6	17	3,679.6	10,466	216.2	12	3,612.3	
582 × 55,515 [rail582]	–	–	40,744	380.2	–	–	–	–	–	–
	0.5	5	27,332	254.2	79	328.4	23,080	224.1	10	436.4
	0.5	10	24,699	249.1	58	633.8	27,098	271.5	14	798.9
	0.5	20	24,825	304.8	51	1,244.0	24,360	325.4	33	1,430.0
	1	5	27,451	264.8	63	369.1	25,349	252.2	17	461.9
	1	10	23,127	218.6	22	717.5	24,111	252.7	22	866.1
	1	20	26,429	322.1	41	1,367.8	25,390	311.2	24	1,527.8
	10	5	26,847	263.2	24	1,165.8	24,984	246.1	16	1,179.3
	10	10	23,065	224.4	23	1,298.0	24,083	246.7	20	1,315.2
10	20	25,355	385.2	30	2,219.1	22,892	302.5	17	2,232.1	
500 × 5,000 [scpnre1]	–	–	34,091	141.1	–	–	–	–	–	–
	0.5	5	16,404	69.3	18	48.2	22,101	80.9	5	147.8
	0.5	10	13,583	58.1	14	107.6	14,111	56.3	8	181.3
	0.5	20	13,277	59.6	13	220.2	16,508	69.7	5	557.2
	1	5	21,868	93.8	8	97.6	21,241	89.6	4	185.5
	1	10	13,583	59.3	14	108.4	14,615	57.1	3	470.3
	1	20	13,277	60.1	13	220.8	15,336	71.6	4	695.8
	10	5	18,355	67.5	20	71.4	20,365	86.3	3	338.0
	10	10	16,746	62.4	7	244.4	14,013	54.4	6	276.0
10	20	13,129	52.6	8	378.3	14,141	61.9	5	588.2	
1,000 × 10,000 [scpnrg1]	–	–	216,009	2,591.2	–	–	–	–	–	–
	0.5	5	99,321	1,485.6	8	258.1	100,534	1,466.9	3	671.3
	0.5	10	91,055	1,239.0	14	282.8	88,183	1,187.1	6	669.2
	0.5	20	71,980	951.1	29	282.8	73,070	1,071.3	7	1,137.4
	1	5	94,423	1,281.0	24	97.3	100,009	1,478.2	6	345.5
	1	10	87,849	1,245.1	17	246.5	89,298	1,185.7	9	450.8
	1	20	66,153	882.6	15	538.9	74,977	1,003.3	4	1,985.0
	10	5	100,213	1,413.1	13	233.0	94,269	1,342.3	4	642.3
	10	10	83,007	1,179.3	3	1,513.7	88,839	1,230.5	4	1,124.5
10	20	75,001	1,048.1	6	1,456.0	73,406	967.2	4	2,058.5	

¹ Notations as in Table 1. Further, mult is the multiple of the number of degenerate basic variables that gives how many additional randomly chosen edge directions are included in RDDFP.

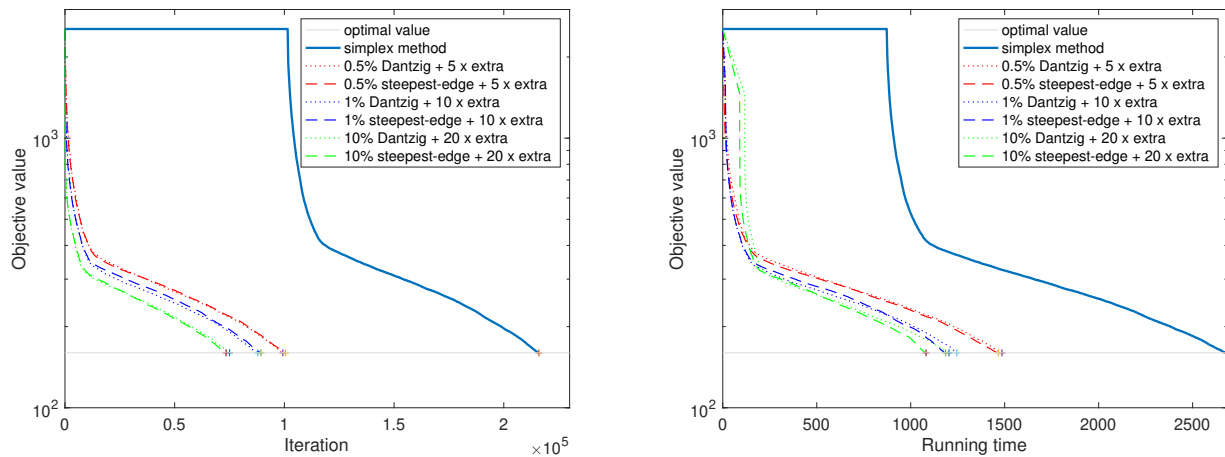


Figure 3. Objective value versus simplex iteration and running time, respectively, for the instance `scpnrg1`

It is also for the degenerate instances possible to reduce both the number of iterations and the running time significantly, but the strategies that give the best results are highly problem dependent. However, the strategy of taking the 1% best edge directions based on the Dantzig selection and adding 5 times the number of degenerate basic variables randomly chosen edge directions, shows a relatively consistent good performance for all the instances except `rail507`. For the instances `scpnre1` and `scpnrg1` it is possible to obtain even better performance by simply using all edge directions with negative reduced costs for constructing the external column. This is because these instances contain relatively few variables compared to the `rail` instances.

As can be seen in Figure 3, the degeneracy causes the standard simplex method to stall for many iterations at the initial basic feasible solution, while our method can find a feasible direction of descent and immediately move to an improved feasible solution, even when the restricted RDDFP includes relatively few edge directions. This advantage of our method can of course be very beneficial when dealing with degenerate linear programs.

We conclude the experimental results by comparing the use of the restricted RDDFP for constructing feasible directions and external columns with the use of the *ad hoc* weighting strategies M2 and M3 [6, 7]. Instead of solving the restricted RDDFP with the quadratic programming solver, we then apply the M2 and M3 weighting strategies to the columns included in the restricted RDDFP; the M2 and M3 rules will then produce approximate solutions to the restricted RDDFP, at a relatively low computational cost.

The most important outcome from this experiment is that both the strategies M2 and M3 always fail to produce feasible directions at the initial basis, for all the instances given in Table 3 and for all the values of `port` and `mult` used there. This shows that in the case of degeneracy, these strategies are very fragile and can not be relied upon, which is notable since many real-life linear programs are (heavily) degenerate. In contrast, the restricted RDDFP, constructed as described above, is always able to find feasible directions, for all the instances and all the choices of parameter values. These results show that the degeneracy-breaking constraints in RDDFP are instrumental in finding feasible directions.

Preliminary experiments indicated that for nondegenerate instances the performance of the restricted RDDFP versus rules M2 and M3 depends of the shape of the linear program, which prompted us to study instances with $n \gg m$. In Table 4 we give results for five additional nondegenerate instances. For

three out of these, $n \gg m$ holds. A conclusion from this experiment (and others, not reported here) is that the rules M2 and M3 both perform well compared to using the restricted RDDFP whenever m and n are of the same magnitude. Further, there is in this case no consistent or significant difference in performance between M2 and M3. However, when n is much larger than m , the use of the restricted RDDFP outperforms both the rules M2 and M3; this conclusion is notable, since it is most common in real-life linear programs that n is much larger than m .

Table 4. Comparison between using the restricted RDDFP called RRDDFP for constructing feasible directions and external columns and using the *ad hoc* weighting strategies M2 and M3 on nondegenerate instances. Other notations as in Table 1

Problem size $m \times n$	Dantzig selection								Steepest-edge selection					
	Parameters		RRDDFP		M2		M3		RRDDFP		M2		M3	
	port	max	iter	time	iter	time	iter	time	iter	time	iter	time	iter	time
1,000 × 3,000	-	-	47,647	828.8	-	-	-	-	-	-	-	-	-	-
	5	∞	28,172	448.6	26,285	433.5	24,289	400.3	21,768	334.5	20,807	333.3	19,746	311.1
	5	1,000	27,689	454.4	27,513	444.7	21,878	369.1	22,369	373.3	20,812	350.6	19,955	344.7
	5	exit	28,211	462.7	27,479	469.9	22,708	400.6	22,282	389.3	20,682	337.3	19,996	369.5
	10	∞	26,121	414.9	21,049	326.1	17,820	270.8	25,480	403.2	18,456	291.1	16,085	250.1
	10	1,000	27,304	426.8	20,554	325.8	19,019	321.8	23,337	396.4	17,320	290.9	18,204	314.3
	10	exit	25,568	412.7	20,909	336.7	19,304	336.8	24,745	399.7	17,106	276.1	17,423	299.7
	20	∞	24,106	383.3	21,010	328.1	22,111	347.1	19,297	298.7	22,621	365.2	22,005	354.1
	20	1,000	23,172	381.6	21,306	341.4	24,886	427.2	18,761	308.9	19,877	324.7	19,811	344.6
	20	exit	22,519	361.7	20,193	357.3	22,632	403.6	18,318	297.1	21,597	372.6	22,566	401.5
1,000 × 6,000	-	-	61,471	1,646.4	-	-	-	-	-	-	-	-	-	-
	5	∞	39,439	1,000.1	41,864	1,067.4	43,331	1,106.4	33,026	829.5	32,781	819.9	35,684	874.3
	5	1,000	37,234	926.4	39,006	966.5	42,845	1,080.2	36,901	988.2	33,587	888.3	34,070	899.1
	5	exit	40,343	1,016.4	42,196	1,117.4	45,559	1,159.3	32,924	824.7	33,073	865.5	35,450	918.2
	10	∞	39,725	1,024.7	37,020	928.8	40,261	1,016.0	33,385	852.3	34,718	868.1	36,513	902.1
	10	1,000	39,480	1,002.6	36,535	959.9	39,540	995.9	30,552	811.1	34,799	936.1	36,494	937.5
	10	exit	38,373	985.1	36,596	976.6	37,365	957.9	31,931	806.6	33,514	879.7	35,272	902.6
	20	∞	36,806	931.8	42,946	1,088.9	45,293	1,204.4	36,453	949.4	43,414	1,125.2	43,531	1,091.9
	20	1,000	39,877	1,024.5	40,638	1,031.8	40,803	1,033.4	35,221	956.9	40,997	1,115.8	37,404	948.7
	20	exit	36,385	920.6	42,666	1,221.5	41,209	1,082.6	34,432	884.3	40,290	1,179.9	45,075	1,235.7
500 × 5,000	-	-	17,182	144.4	-	-	-	-	-	-	-	-	-	-
	5	∞	11,027	91.5	12,339	103.1	13,064	104.6	10,140	82.9	9,251	74.9	9,354	73.4
	5	1,000	12,398	103.6	13,057	106.5	13,564	108.2	10,154	85.1	9,289	77.5	9,599	76.2
	5	exit	10,544	87.3	12,444	110.2	13,518	108.1	9,645	78.9	9,239	79.2	9,384	74.6
	10	∞	10,814	88.8	13,383	112.2	12,335	98.6	10,251	83.4	12,128	98.8	10,422	116.8
	10	1,000	11,976	98.7	13,274	110.2	12,722	102.1	10,606	88.3	10,380	85.5	10,601	85.7
	10	exit	9,999	81.5	13,447	122.4	13,035	114.7	9,947	79.3	12,320	105.1	10,397	84.5
	20	∞	11,530	95.8	13,591	114.4	13,229	106.7	11,091	91.5	14,172	121.7	12,741	105.2
	20	1,000	11,630	97.7	13,255	112.1	13,118	106.8	10,545	88.5	14,310	127.2	12,444	101.0
	20	exit	11,222	94.6	12,470	109.2	13,643	118.4	10,442	84.3	12,593	129.6	13,517	116.2
500 × 10,000	-	-	16,515	241.6	-	-	-	-	-	-	-	-	-	-
	5	∞	12,643	183.8	13,930	199.4	14,750	214.9	10,787	157.3	12,638	179.1	12,419	178.8
	5	1,000	11,895	169.6	14,276	200.7	14,781	208.0	10,570	150.1	12,183	174.3	11,441	194.1
	5	exit	12,852	190.1	14,598	253.6	14,798	226.5	10,959	152.8	12,155	173.9	12,226	173.1
	10	∞	11,033	155.8	12,097	171.2	13,480	192.4	10,262	145.5	13,274	189.9	13,286	189.2
	10	1,000	12,409	178.5	13,353	189.8	13,359	187.7	11,151	160.9	12,515	177.5	12,567	176.3
	10	exit	11,330	161.3	12,592	200.9	13,272	214.2	9,932	141.6	12,775	187.1	12,932	191.9
	20	∞	11,800	168.8	14,560	214.3	13,778	198.1	11,572	165.5	14,589	211.8	14,411	205.2
	20	1,000	11,602	168.1	14,484	205.7	13,138	184.8	10,682	153.3	13,122	186.5	13,764	193.5
	20	exit	11,433	164.3	14,778	245.4	14,478	216.9	11,390	164.2	13,296	203.3	14,964	229.8
500 × 20,000	-	-	17,065	500.2	-	-	-	-	-	-	-	-	-	-
	5	∞	13,118	377.4	15,950	446.7	16,473	469.7	13,928	393.8	14,331	403.7	13,900	392.8
	5	1,000	13,649	460.9	15,738	440.1	16,241	454.5	12,751	358.5	15,049	423.2	14,564	409.4
	5	exit	13,091	368.1	16,395	511.1	16,520	502.1	13,512	377.6	13,875	403.6	14,177	402.7
	10	∞	12,894	365.8	16,436	469.0	15,791	442.5	12,444	351.4	16,202	465.1	16,703	469.9
	10	1,000	13,983	393.5	17,149	490.6	17,242	480.2	12,156	342.9	15,287	429.1	16,102	458.2
	10	exit	12,849	370.2	17,052	557.6	16,983	500.1	12,253	349.7	15,128	456.6	14,940	431.8
	20	∞	13,471	421.8	16,980	485.6	17,368	492.8	13,038	404.2	15,919	456.6	16,915	482.7
	20	1,000	13,034	377.6	16,827	475.2	17,175	481.5	13,271	388.3	16,644	474.2	16,334	458.8
	20	exit	13,898	420.4	16,093	518.5	17,205	517.3	12,551	417.0	15,591	513.8	16,608	505.2

5. Conclusions

We have derived a simplex-type feasible direction method for linear programming. The directions are steepest in the space of all variables and they are found by solving a quadratic direction-finding problem. A feature of this problem is that it includes degeneracy-breaking constraints, which prevents stalling at the current feasible solution. In practice, it is preferable to consider a restricted direction-finding problem, which gives an approximately steepest feasible direction. The method is easily embedded within the standard simplex method by using external columns. Our linear programming method allows many computational strategies, especially regarding the construction and solution of the restricted direction-finding problem and the strategy for replacing ordinary simplex pivots with pivots on external columns.

The presented results, and also our experience from experiments not reported here, indicate that the use of approximate steepest feasible directions can considerably reduce both the number of simplex iterations and the total running time. It is however necessary to find a proper trade-off between the computational burden of creating advantageous external columns and the reductions in simplex iterations and running times that they may lead to, both with respect to the size of the restricted direction-finding problem and the frequency of generating external columns. Further, this trade-off depends of the characteristics of the linear program at hand. The general observation is that the direction-finding problems should be neither too restricted nor too large, and that external columns should be generated seldomly. If using too restricted or too large direction-finding problems, or if generating external columns frequently, then the overall performance can instead actually get worse than that of the standard simplex method.

The approach presented here is related to the ones in [6, 7] and [8, 19], which both also use auxiliary primal variables for following a feasible direction. However, these two works do not use any direction-finding optimization problem, but rely on *ad hoc* rules for constructing feasible directions, for example, based on reduced costs. We have compared our approach to such *ad hoc* rules. We observe that our approach performs better when there are many more columns than constraints in the linear program and that our approach can effectively handle degeneracy, while the *ad hoc* rules can fail in this situation. These observations are notable since real-life linear programs typically have much more columns than constraints and are degenerate.

The computational results are promising for further investigation of our feasible direction approach. This includes the study of various computational strategies and the design of tailored algorithms for fast, approximate solution of restricted versions of the direction-finding problem RDDFP, for example, algorithms based on the projected Newton strategy [4]. Further, the ability of our method to find a feasible direction of descent from a degenerate extreme point, so that the simplex method can escape from such a point in a single pivot, is worth further consideration.

As is well known (see, e.g., [15]), the performance of the standard simplex method is sensitive to problem scaling (column and row scaling). An interesting research topic is therefore to investigate if and how the performance of our simplex-type feasible direction approach is affected by problem scaling.

A fundamental research question is the extension of our simplex-type feasible direction method to a column generation setting, that is, structured linear programs that contain huge numbers of variables that can be found by solving column generation problems. In such an extension it would be necessary to apply a column generation approach to build a direction-finding problem.

Acknowledgement

The authors gratefully acknowledge the suggestion of a reviewer to compare our approach with the *ad hoc* rules previously suggested. This led to a significant improvement of this work.

References

- [1] ARSHAM, H. A hybrid gradient and feasible direction pivotal solution algorithm for general linear programs. *Applied Mathematics and Computation* 188, 1 (2007), 596–611.
- [2] BAZARAA, M. S., SHERALI, H. D., AND SHETTY, C. M. *Nonlinear Programming: Theory and Algorithms*. 3rd edition, John Wiley & Sons, 2006.
- [3] BEASLEY, J. E. OR-Library: distributing test problems by electronic mail. *Journal of the Operational Research Society* 41, 11 (1990), 1069–1072.
- [4] BERTSEKAS, D. P. Projected Newton methods for optimization problems with simple constraints. *SIAM Journal on Control and Optimization* 20, 2 (1982), 221–246.
- [5] BORGWARDT, K. H., AND HUHN, P. A lower bound on the average number of pivot-steps for solving linear programs valid for all variants of the simplex-algorithm. *Mathematical Methods of Operations Research* 49 (1999), 175–210.
- [6] EISELT, H. A., AND SANDBLOM, C.-L. External pivoting in the simplex algorithm. *Statistica Neerlandica* 39, 4 (1985), 327–341.
- [7] EISELT, H. A., AND SANDBLOM, C.-L. Experiments with external pivoting. *Computers and Operations Research* 17, 4 (1990), 325–332.
- [8] FATHI, Y., AND MURTY, K. G. Computational behavior of a feasible direction method for linear programming. *European Journal of Operational Research* 40, 3 (1989), 322–328.
- [9] FORREST, J. J., AND GOLDFARB, D. Steepest-edge simplex algorithms for linear programming. *Mathematical Programming* 57, 1-3 (1992), 341–374.
- [10] FUKUDA, K., AND TERLAKY, T. On the existence of a short admissible pivot sequence for feasibility and linear optimization problems. *Pure Mathematics and Applications* 10, 4 (1999), 431–447.
- [11] GOLDFARB, D., AND REID, J. K. A practicable steepest-edge simplex algorithm. *Mathematical Programming* 12 (1977), 361–371.
- [12] GONDZIO, J. Another simplex-type method for large scale linear programming. *Control and Cybernetics* 25, 4 (1996), 739–760.
- [13] HARRIS, P. M. J. Pivot selection methods of the Devex LP code. *Mathematical Programming* 5, 1 (1973), 1–28.
- [14] KLEE, V., AND MINTY, G. J. How good is the simplex algorithm? In *Inequalities III* (1972), O. Shisha, Ed., Academic Press, pp. 159–175.
- [15] LARSSON, T. On scaling linear programs—some experimental results. *Optimization* 27, 4 (1993), 355–373.
- [16] MAROS, I. *Computational Techniques of the Simplex Method*. Kluwer Academic Publishers, 2003.
- [17] MITRA, G., TAMIZ, M., AND YADEGAR, J. Experimental investigation of an interior search method within a simplex framework. *Communications of the ACM* 31, 12 (1988), 1474–1482.
- [18] MURTY, K. G. *Linear Programming*. 2nd edition. Wiley, 1983.
- [19] MURTY, K. G., AND FATHI, Y. A feasible direction method for linear programming. *Operations Research Letters* 3, 3 (1984), 121–127.
- [20] WOLDE, B. C., AND LARSSON, T. A steepest feasible direction extension of the simplex method. In *Operations Research Proceedings 2019, Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), Dresden, Germany, September 4-6, 2019* (Cham, 2020), J. S. Neufeld, U. Buscher, R. Lasch, D. Möst, and J. Schönberger, Eds., Springer, pp. 113–121.