*Les Cahiers du GERAD* ISSN: 0711–2440

# Positive Edge: A Pricing Criterion for the Identification of Non-Degenerate Simplex Pivots

V. Raymond, F. Soumis
A. Metrane, J. Desrosiers

G–2010–61

October 2010

# Positive Edge: A Pricing Criterion for the Identification of Non-Degenerate Simplex Pivots

**Vincent Raymond**
**François Soumis**

*GERAD & École Polytechnique de Montréal*
*C.P. 6079, succ. Centre-Ville*
*Montréal (Québec) Canada, H3C 3A7*
vincent.raymond@polymtl.ca; francois.soumis@gerad.ca

**Abdelmoutalib Metrane**

*GERAD*
*3000 chemin de la Côte-Sainte-Catherine*
*Montréal (Québec) Canada, H3T 2A7*
abdelmoutalib.metrane@gerad.ca

**Jacques Desrosiers**

*GERAD & HEC Montréal*
*3000 chemin de la Côte-Sainte-Catherine*
*Montréal (Québec) Canada, H3T 2A7*
jacques.desrosiers@hec.ca

October 2010

*Les Cahiers du GERAD*

G–2010–61

## Abstract

The *positive edge* is a new pricing rule for the primal simplex: it identifies, with a probability error less than or equal to $2^{-30}$ in single precision binary floating-point format and $2^{-62}$ in double precision format, variables allowing for non-degenerate pivots. These are identified directly from a short calculation on the original coefficients of the constraint matrix. The complexity is the same as for the computation of the reduced cost. If such a variable has a negative reduced cost, it strictly improves the objective function value when entered into the basis.

The preliminary computational experiments made with CPLEX show its high potential. We designed a simple algorithm using two external procedures: one identifies variables that allow for non-degenerate pivots while the other identifies variables with negative reduced cost. These are sent to the primal simplex algorithm of CPLEX. It has been tested on fourteen medium-sized aircraft fleet assignment instances (5000 constraints and 25 000 variables), two large-scale manpower planning problems (100 000 constraints and 450 000 variables), and nine PDS instances from the Mittelmann library. All these problems are highly degenerate. On the first group, our algorithm is 7.4 times faster than CPLEX on average and the number of pivots is almost reduced by a factor 2. On the second and third groups, it is 50% faster and the number of pivots is decreased by 2.4 and 3.6, respectively. It has also been tested on Fome12 and Fome13 from the Mittelmann library. For these two highly dense problems, our simple implementation failed. The integration of the positive edge rule within a primal simplex code should prevent such cases by eliminating the external procedures and taking advantage of partial pricing strategies.

**Key Words:** linear programming, primal simplex, degeneracy.

## Résumé

Le *positive edge* est un nouveau critère d'entrée pour l'algorithme primal du simplexe. Il identifie, avec une probabilité d'erreur d'au plus $2^{-30}$ en format point flottant simple précision et $2^{-62}$ en format double précision, les variables donnant lieu à des pivots non dégénérés. Elles sont identifiées directement à partir d'un calcul simple sur les données originales de la matrice de contraintes. La complexité de calcul est la même que pour celle du coût réduit. Si une telle variable a un coût réduit négatif, elle améliore strictement la fonction objectif lorsqu'elle entre dans la base.

Les résultats préliminaires obtenus avec CPLEX démontre son énorme potentiel. Nous avons élaboré un algorithme simple faisant appel à deux procédures externes : l'une identifie les variables donnant lieu à des pivots non dégénérés alors que la seconde identifie les variables de coût réduit négatif. Les variables choisies sont envoyées au simplexe primal de CPLEX. Les tests ont été réalisés sur quatorze problèmes d'affectation d'une flotte d'avions (5000 constraints and 25 000 variables), deux instances de planification de la main d'oeuvre (100 000 constraints and 450 000 variables), et neuf problèmes PDS de la librairie Mittelmann. Tous ces problèmes possèdent des solutions hautement dégénérées. Pour le premier groupe, notre algorithe est 7.4 fois plus rapide que CPLEX en moyenne et le nombre de pivots est preque réduit par un facteur 2. Pour les second et troisième groupes, notre implantation est 50% plus rapide et le nombre de pivots est réduit par respectivement 2.4 et 3.6. Nous l'avons également évaluée sur Fome12 et Fome13 de la librairie Mittelmann. Pour ces deux instances très denses, notre implantation trop simple n'est pas performante. L'intégration de la règle du *positive edge* dans un code de simplexe primal devrait empêcher de telles situations de se produire en élinimant les deux procédures externes et en prenant avantages des stratégies d'évaluation partielle des coûts réduits.

**Mots clés :** programmation linéaire, simplexe primal, dégénérescence.

**Acknowledgments:** The work of François Soumis and Jacques Desrosiers was supported by the Canadian Natural Science and Engineering Research Council (CRSNG) and le Fonds québécois de la recherche sur la nature et les technologies (FQRNT).

## 1 Introduction

Consider the following linear programming problem (LP) in standard form

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to:} \quad & Ax = b, \ x \geq 0, \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ is the vector of decision variables, $c \in \mathbb{R}^n$ is the cost vector, $A \in \mathbb{R}^m \times \mathbb{R}^n$ is the constraint matrix of full rank, and $b \in \mathbb{R}^m$ is the right-hand side vector. We are particularly interested in problems for which the various basic solutions are highly degenerate, that is, for which the number of non-zero variables is much less than $m$, the size of the basis. In that case, the primal simplex algorithm (Dantzig, 1949) is likely to encounter degenerate pivots and possibly to cycle. To avoid cycling, several pivot rules and right-hand side perturbation methods have been proposed, e.g., Charnes (1952); Wolfe (1963); Bland (1977); Fukuda (1982); Ryan and Osborne (1988). However, these rules and methods do not strongly improve the performance of the primal simplex algorithm on degenerate problems. Another way of avoiding degenerate pivots is by using the steepest edge criterion (Forrest and Goldfarb, 1992). This criterion computes the improvement of the cost function for possible entering variables, that is, the step size multiplied by the reduced cost. Hence, if one exists, it selects a variable with a non-degenerate pivot. However, the steepest edge criterion requires a significant amount of CPU time.

To improve the solution time on degenerate problems, Pan (1998) proposes the use of a *reduced problem* with a smaller number of constraints and variables. His method starts with an initial basic solution and identifies its $p$ non-zero basic variables. Constraints are split in two: set $P$ where the basic variables takes a positive value and set $Z$ where the basic variables are zero. Variables are also split in two sets. *Compatible* variables are those for which all values are zero in the updated simplex tableau for constraint indexes in $Z$, other variables are said to be *incompatible* – formal definitions are given in Section 2. The $m - p$ constraints in $Z$ are temporarily removed to leave a smaller constraint matrix with only $p$ rows. To preserve feasibility, incompatible variables are also removed to form the reduced problem. Since the $p \times p$ basis of the reduced problem is non-degenerate, that is, the number of non-zero variables is $p$, the next pivot is automatically non-degenerate. The resulting reduced problem is solved to optimality over the compatible variables and the reduced costs are computed by means of its dual variables. In Pan's method, dual variables of LP corresponding to the $m - p$ eliminated constraints are arbitrarily set to zero. Next, incompatible variables are considered. If such a variable is to become basic, some of the eliminated constraints must be reintroduced in the reduced problem. When compared to his own implementation of the primal simplex algorithm, Pan (1998) reports speed-up factors of 4 to 5.

In a column-generation framework (which can be seen as a primal simplex approach), Elhallaoui et al. (2005) propose a *dynamic constraint aggregation* (DCA) method for the solution of the linear relaxation of set partitioning problems. Considering only the $p$ non-zero basic variables, the DCA method identifies identical rows (composed of zeros and ones) of the corresponding columns. In the *constraint aggregation* phase, a single constraint per row-group remains in the reduced problem. Authors show that, once the reduced problem has been solved, the dual variable of a kept constraint is equal to the sum of the dual variables of the corresponding row-group. A full set of dual variables is recovered by distributing adequately the values of the dual variables of the reduced problem. For set partitioning problems, this is done by solving a shortest-path problem. These dual variables are used to price out the generated columns, allowing for updates of the constraint aggregation. To improve their method, the authors developed the *multi-phase dynamic constraint aggregation* (MPDCA) method (Elhallaoui et al., 2010). The goal of MPDCA is to introduce in the reduced problem the incompatible variables in the order of their so-called incompatibility number. On a set of large-scale bus driver scheduling problems, MPDCA reduces the solution time by a factor of more than 23 over the classical column-generation method.

The *Improved Primal Simplex* (IPS) method of Elhallaoui et al. (2007) combines ideas from the reduced problem of Pan (1998) and from dynamic constraint aggregation of Elhallaoui et al. (2005) to solve linear programming problems. Considering only the $p$ non-zero basic variables, IPS identifies a set of $p$ rows that are linearly independent and removes from the reduced problem all the other $m-p$ constraints. As in Pan (1998), the reduced problem is solved using the compatible variables only. Next, a *complementary problem* is constructed and solved to prove that the current solution of the reduced problem is optimal for LP, otherwise it selects a set of incompatible variables to be introduced into the current reduced problem. Elhallaoui et al. (2007) show that when the solution of the reduced problem is not optimal for LP, the re-optimization after adding all the incompatible variables of the chosen set strictly decreases the objective function value. Indeed, they show that in that case, there exists a convex combination of the selected incompatible variables that is compatible with respect to the reduced problem, hence with a strictly positive step size. The complementary problem contains all the incompatible variables and its coefficient matrix is created at the same time as the reduced problem. As explained in Section 2, the reduced and the complementary problems are built according to a modification of the original constraint matrix. Indeed, this modified matrix (which is the result of an updated simplex tableau) is obtained by multiplying $A$ by the current inverse of the basis matrix. The complexity of computing this modified matrix for the identification of the compatible variables is $\mathrm{O}(m^2 n)$ (see Section 3). The computational results of Raymond, Soumis, and Orban (2010) show that, on medium-sized instances ($m \approx 5000$, $n \approx 25\,000$), IPS is faster than the primal simplex algorithm of CPLEX by factors ranging from 5 to 20. However, on large-scaled problems ($m \approx 100\,000$, $n \approx 450\,000$), constructing the reduced and the complementary problems is too costly compared to the primal simplex algorithm itself.

In this paper, as in IPS, priority is given to non-degenerate pivots. However, compatible variables that form the reduced problem are identified directly from the original constraint matrix $A$ instead of from the modified matrix obtained by multiplying it by the inverse of the basis. The new criterion is called *positive edge*. Determining which variables are compatible is done in $O(mn)$, i.e., $O(m)$ for each variable, the same complexity as for the reduced cost computation of such a variable. Obviously, as in IPS, one might have to execute some degenerate pivots to reach optimality.

The paper is organized as follows. The next section Section 2 presents the reduced problem and the compatibility concept of Pan (1998). In Section 3, we elaborate on the positive edge criterion with a new definition for compatibility and study its complexity. Reliability of it is proven is Section 4. We propose an integration to the primal simplex algorithm in Section 5 based on a two-dimensional selection rule for negative reduced cost variables. In Section 6, we propose a simple implementation that uses two external procedures for the selection of the variables sent to the primal simplex algorithm of a commercial code. Computational experiments with CPLEX are reported in Section 7 and show the high potential of the positive edge criterion for highly degenerate instances. Finally, the conclusion follows in Section 8.

**Notation.** If $x \in \mathbb{R}^n$ and $\mathrm{C} \subset \{1, \ldots, n\}$ is an index set, $x_{\mathrm{C}}$ denotes the sub-vector of $x$ indexed by C. Similarly, if $A$ is an $m \times n$ matrix, we denote by $A_{\mathrm{C}}$ the $m \times |\mathrm{C}|$ matrix whose columns are indexed by C. If $\mathrm{I} = \{1, \ldots, n\} \setminus \mathrm{C}$, then $x = (x_{\mathrm{C}}, x_{\mathrm{I}})$ even though the indexes in C and I may not appear in that order. Similarly, we use an upper index on a matrix to refer to a subset of its rows. Finally, define $\mathbb{R}_0 = \mathbb{R} \setminus \{0\}$ to be the set of non-zero real numbers.

## 2 The Reduced Problem

In this section, we establish the notation and summarize the construction and the key properties of the reduced problem proposed by Pan (1998) and largely used for IPS in Elhallaoui et al. (2007) and Raymond, Soumis, and Orban (2010).

Assume $m < n$ and let $B \subset \{1, ..., n\}$ be the index set of the current basic variables, i.e., the basis matrix for LP is given by $A_{\mathrm{B}}$. Define $Q = A_{\mathrm{B}}^{-1}$ and let $x_{\mathrm{B}} = Qb = \bar{b}$ be a degenerate feasible solution with $1 \le p < m$ non-zero variables. Let $\mathrm{P} = \{i \in \{1, ..., m\} \mid \bar{b}_i > 0\}$ be the index set of the $p$ rows where the positive (or non-degenerate) variables appear as basic and $\mathrm{Z} = \{i \in \{1, ..., m\} \mid \bar{b}_i = 0\}$ be the index set of the $m-p$ rows where the degenerate basic variables (taking value zero) appear. Make the following partition of the rows of the inverse of the basis:

$$Q = \begin{bmatrix} Q^{\mathrm{P}} \\ Q^{\mathrm{Z}} \end{bmatrix},$$

where $Q^{\text{z}}$, of size $(m-p)\times m$, is called the *compatibility matrix.* Therefore

$$\bar{b} = \left[\ \bar{b}_i\ \right]_{i\in\{1,\ldots,m\}} = \begin{bmatrix} Q^{\text{P}}b \\ Q^{\text{Z}}b \end{bmatrix} = \begin{bmatrix} Q^{\text{P}}b \\ 0 \end{bmatrix},$$

where $Q^{\text{P}}b > 0$ gives the value of the non-degenerate basic variables. Let $A_j = \left[\ a_{ij}\ \right]_{i\in\{1,\ldots,m\}}$ be the $j^{th}$ column of $A$. We begin with the following definition.

**Definition 1** *Variable* $x_j, j \in \{1,...,n\}$ *is* compatible with respect to $Q^{\text{z}}$ *if and only if* $Q^{\text{z}}A_j = 0$, *i.e.,* $\bar{A}^{\text{z}}_j = 0$.

In other words, a variable $x_j$ is compatible with respect to matrix $Q^{\text{z}}$ if and only if, in the simplex tableau obtained by multiplying the original system of constraints by the inverse $Q$ of the basis, all $m-p$ components of the updated column-vector $\bar{A}_j = QA_j \in \mathbb{R}^m$ are zero in the row-set Z, that is:

$$\bar{A}_j = \left[\ \bar{a}_{ij}\ \right]_{i\in\{1,\ldots,m\}} = \begin{bmatrix} \bar{A}^{\text{P}}_j \\ \bar{A}^{\text{Z}}_j \end{bmatrix} = \begin{bmatrix} Q^{\text{P}}A_j \\ Q^{\text{Z}}A_j \end{bmatrix} = \begin{bmatrix} Q^{\text{P}}A_j \\ 0 \end{bmatrix}.$$

When a variable $x_j$ is compatible, a natural extension is to say that its associated column $A_j$ is compatible. A variable $x_j$ for which $\bar{A}^{\text{z}}_j = Q^{\text{z}}A_j \neq 0$ is said to be incompatible with respect to $Q^{\text{z}}$. Observe that positive basic variables are compatible whereas all degenerate ones are not.

Consider a non-basic compatible variable $x_j, j \notin B$ with a negative reduced cost that is selected to enter into the basis. Because $\bar{a}_{ij} = 0, \forall i \in \text{Z}$, the step size given by the usual ratio-test can be computed only on the row-set P, that is:

$$\min_{i\in\{1,\ldots,m\}} \{\frac{\bar{b}_i}{\bar{a}_{ij}} | \bar{a}_{ij} > 0\} = \min_{i\in\text{P}} \{\frac{\bar{b}_i}{\bar{a}_{ij}} | \bar{a}_{ij} > 0\}.$$

Since $\bar{b}_i > 0, \forall i \in \text{P}$, the step size for such a compatible variable $x_j$ with a negative reduced cost is greater than zero and the objective function strictly improves, unless $\bar{a}_{ij} \leq 0, \forall i \in \text{P}$, in which case LP is unbounded.

Let $\text{C} \subset \{1,\ldots,n\}$ and $\text{I} = \{1,\ldots,n\} \setminus \text{C}$ denote the index sets of the compatible and incompatible variables, respectively. Partition accordingly the vector of variables $x = (x_{\text{C}}, x_{\text{I}})$, the cost vector $c = (c_{\text{C}}, c_{\text{I}})$, and the constraint matrix $A = (A_{\text{C}}, A_{\text{I}})$. Upon pre-multiplying by $Q$ the system of constraints of LP, we obtain $QAx = Qb$, which can be rewritten as

$$\begin{bmatrix} Q^{\text{P}}Ax \\ Q^{\text{Z}}Ax \end{bmatrix} = \begin{bmatrix} Q^{\text{P}}A_{\text{C}}x_{\text{C}} + Q^{\text{P}}A_{\text{I}}x_{\text{I}} \\ Q^{\text{Z}}A_{\text{I}}x_{\text{I}} \end{bmatrix} = \begin{bmatrix} Q^{\text{P}}b \\ 0 \end{bmatrix}.$$

By definition vector $x_{\text{C}}$ is compatible with respect to $Q^{\text{z}}$ and we have $Q^{\text{z}}A_{\text{C}} = 0$. Therefore, the original problem LP can equivalently be rewritten as

$$\begin{array}{lrcrcr} \text{minimize} & c_{\text{C}}^{\top}x_{\text{C}} & + & c_{\text{I}}^{\top}x_{\text{I}} & & \\ \text{subject to:} & Q^{\text{P}}A_{\text{C}}x_{\text{C}} & + & Q^{\text{P}}A_{\text{I}}x_{\text{I}} & = & Q^{\text{P}}b \\ & & & Q^{\text{Z}}A_{\text{I}}x_{\text{I}} & = & 0 \\ & x_{\text{C}}, & & x_{\text{I}} & \geq & 0. \end{array}$$

This reformulation of LP is interesting in several aspects. Naturally, its solution is given in terms of $x = (x_{\text{C}}, x_{\text{I}})$, that is, a mix of compatible and incompatible variable values. By definition, compatible variables $x_{\text{C}}$ already satisfy $Q^{\text{z}}A_{\text{C}} = 0$. Amazingly, incompatible variables $x_{\text{I}}$ must also satisfy a similar expression:

$$Q^{\text{Z}}(A_{\text{I}}x_{\text{I}}) = 0, \quad x_{\text{I}} \geq 0.$$

Indeed, in any feasible or optimal solution, the column-vector $(A_{\text{I}}x_{\text{I}})$, a non-negative combination of incompatible columns, is compatible with respect to $Q^{\text{z}}$. This is one of the main properties exploited in IPS by

Elhallaoui et al. (2007) and Raymond, Soumis, and Orban (2010) for taking advantage of degeneracy in linear programming. Upon imposing $x_{\mathrm{I}} = 0$, we obtain the *reduced problem* (RP):

$$\begin{aligned} &\text{minimize} && c_{\mathrm{C}}^{\top} x_{\mathrm{C}} \\ &\text{subject to:} && Q^{\mathrm{P}} A_{\mathrm{C}} x_{\mathrm{C}} = Q^{\mathrm{P}} b,\ x_{\mathrm{C}} \geq 0. \end{aligned} \tag{2}$$

RP contains $p$ constraints and is potentially much smaller than LP in terms of the number of rows. Moreover it only depends on the compatible variables $x_{\mathrm{C}}$. Any compatible variable can enter into a basis for RP without violating the constraints of LP that have been omitted. This is why we also say that a variable $x_j$ or its column-vector $A_j$, compatible [incompatible] with respect to $Q^{\mathrm{Z}}$, is compatible [incompatible] with respect to the reduced problem RP.

Finally, note that by construction, if $x_{\mathrm{C}}$ is feasible for RP, then $x = (x_{\mathrm{C}}, 0)$ is also feasible for LP. Obviously, one might have to consider the incompatible variables $x_{\mathrm{I}}$ and execute some degenerate pivots to reach optimality of LP.

## 3 The Positive Edge Criterion

The identification of all the compatible variables using Definition 1 has a complexity $O(m^2 n)$ since one needs to compute $Q^{\mathrm{Z}} A$, see Section 3.2. For large-scale problems, this method may be more costly, in terms of CPU time, than the solution of LP with the simplex algorithm.

To identify all the non-basic variables that are compatible with respect to $Q^{\mathrm{Z}}$, we developed the *positive edge* criterion. The proposed rule allows to know almost surely if a variable $x_j$ can enter the basis with a positive value without explicitly computing neither $Q^{\mathrm{Z}} A_j$ nor the step size given by the ratio-test. If such a variable has a negative reduced cost, the objective function of LP strictly decreases when it enters into the basis since its step size computed over the index set P is greater than zero. In the following, we show how the positive edge criterion identifies compatible variables directly from a short calculation on the original coefficients of the constraint matrix $A$.

### 3.1 A Compatibility Definition Based on a Scalar Product

From Definition 1, we know that non-degenerate basic variables are compatible with respect to $Q^{\mathrm{Z}}$. Let $v \in \mathbb{R}_0^{m-p}$ be a vector of $m - p$ non-zero components and compute $w^{\top} = v^{\top} Q^{\mathrm{Z}}$, a row-vector in $\mathbb{R}^m$. If a non-basic variable $x_j, j \notin B$ is compatible with respect to $Q^{\mathrm{Z}}$, i.e., $\bar{A}_j^{\mathrm{Z}} = 0$, then necessarily $v^{\top} \bar{A}_j^{\mathrm{Z}} = 0 \in \mathbb{R}$, hence the scalar product between $w$ and the original column-vector $A_j$ is also null:

$$v^{\top} \bar{A}_j^{\mathrm{Z}} = v^{\top} Q^{\mathrm{Z}} A_j = w^{\top} A_j = 0.$$

Can we choose $w$, in fact $v$ with the appropriate properties, such that this simple scalar product condition becomes sufficient on the original data columns? That is, if $w^{\top} A_j = 0$ then $\bar{A}_j^{\mathrm{Z}} = 0$ and variable $x_j, j \notin B$ is compatible with respect to $Q^{\mathrm{Z}}$, or equivalently, if $\bar{A}_j^{\mathrm{Z}} \neq 0$ then $w^{\top} A_j \neq 0$ and variable $x_j, j \notin B$ is incompatible with respect to $Q^{\mathrm{Z}}$. To answer this question, consider a new definition of compatibility based on $w^{\top} A_j$.

**Definition 2** *Let $v \in \mathbb{R}_0^{m-p}$ and $w^{\top} = v^{\top} Q^{\mathrm{Z}}$. A non-basic variable $x_j, j \notin B$ is* compatible with respect to $v^{\top} Q^{\mathrm{Z}}$ *if and only if $w^{\top} A_j = 0$.*

In Section 4, we show that almost surely, when choosing $v$ with specific random properties, if a non-basic variable $x_j$ is incompatible with respect to $v^{\top} Q^{\mathrm{Z}}$, then it is also incompatible with respect to $Q^{\mathrm{Z}}$. Indeed, if $w^{\top} A_j \neq 0, j \notin B$ then $\bar{A}_j^{\mathrm{Z}} \neq 0$ with a probability error smaller than or equal to $2^{-30}$ in single precision binary floating-point format and $2^{-62}$ in double precision format.

### 3.2 Computational Complexity of the Positive Edge

In the following, we discuss the computational complexity of the positive edge criterion versus that of Pan's method. Let $\lambda_A$ be the density of the constraint matrix $A$ and $\ell$ be the number of basic computer cycles needed for a multiplication and an addition. To identify all the non-basic compatible variables with respect to $Q^{\mathrm{z}}$ with Pan's method, we need to compute $QA_j, \forall j \notin B$. This requires the multiplication of the $m \times m$ inverse basis matrix $Q$ by the $n-m$ column-vectors $A_j, j \notin B$. For each coefficient of the resulting vector $QA_j$, a row of $Q$ is multiplied by column $A_j$. This uses $\ell\lambda_A m$ basic computer cycles. To compute the $m(n-m)$ coefficients, we get $m(n-m)\ell\lambda_A m = O(m^2 n)$.

To identify all the non-basic compatible variables with respect to $v^\top Q^{\mathrm{z}}$, we only compute $w^\top A_j$, $\forall j \notin B$. Let $\lambda_{Q^{\mathrm{z}}}$ be the percentage of the non-zero coefficients in the compatibility matrix $Q^{\mathrm{z}}$. Firstly, we need to compute the $m$-dimensional vector $w^\top = v^\top Q^{\mathrm{z}}$, that is, to multiply a vector $v$ of $m-p$ non-zero components by the $(m-p) \times m$ matrix $Q^{\mathrm{z}}$: this takes $m\ell\lambda_{Q^{\mathrm{z}}}(m-p)$ basic cycles. Secondly, we compute the $n-m$ values $w^\top A_j, j \notin B$, that is, we multiply the $m$-vector $w$ by $n-m$ columns of $A$: it takes $(n-m)\ell\lambda_A m$ basic cycles. The total complexity is therefore $[m\ell\lambda_{Q^{\mathrm{z}}}(m-p)] + [(n-m)\ell\lambda_A m] = O(mn)$. Determining if a non-basic variable $x_j$ is compatible or not with respect to $v^\top Q^{\mathrm{z}}$ can be done in $O(m)$, i.e., the same complexity as for the reduced cost computation of that variable, i.e., $\bar{c}_j = c_j - \pi^\top A_j, j \notin B$, where $\pi$ is the vector of dual variables for LP.

On large-scale instances ($m \approx 100\ 000$, $n \approx 450\ 000$), we compared the CPU time needed to identify all the compatible variables with Pan's method and with the positive edge criterion. Pan's method requires 2500 seconds; the positive edge criterion needs only 0.5 seconds.

## 4 Reliability of the Positive Edge Criterion

In this section we study the reliability of the positive edge criterion to identify non-basic compatible and incompatible column-vectors. Essentially, our proposition on the identification of compatible columns with respect to $v^\top Q^{\mathrm{z}}$ could be stated as follows: Assume $\bar{A}_j^{\mathrm{z}} \neq 0$, $j \notin B$. If $V_i, i \in Z$ are independently and identically distributed (i.i.d.) random variables, then

$$P[\sum_{i \in Z} \bar{a}_{ij} V_i = 0 \mid \bar{A}_j^{\mathrm{z}} \neq 0] = 0,$$

that is, the conditional probability that an incompatible variable be identified as a compatible one is null. For random variables with continuous cumulative distribution functions, this is an obvious result. Indeed $\sum_{i \in Z} \bar{a}_{ij} V_i$ is a continuous random variable for which the probability of an atomic event is null. However, with binary representation of real numbers on a computer, one has to make use of discrete probability distributions. The above result becomes almost true with a small probability error. In this section, we measure the size of this error.

Before proving our main propositions, we discuss properties of bit-addition and multiplication of floating-point numbers. We start with the following lemma on bits i.i.d. Bernoulli(0.5), denoted $Bern(0.5)$. Recall that bits are numbered from right to left, the first being numbered zero.

**Lemma 1** *Let $d$ be a positive integer. Consider the addition of $d$-bit numbers $n_1$ and $n_2$. Assume that the bits of $n_1$ are i.i.d. Bern(0.5) whereas the probability distributions of the bits of $n_2$ are unknown. The sum $n_1 + n_2$ is a $(d+1)$-bit number for which the $d$ right-most bits are i.i.d. Bern(0.5) whereas the left-most bit has an unknown probability distribution.*

**Proof.** Observe that the largest value for a $d$-bit number is $2^d - 1$ so that the largest value for $n_1 + n_2$ is $2 \times (2^d - 1) = 2^{d+1} - 2$ which requires $d+1$ bits. Let $0 \leq \alpha_0 \leq 1$ be the probability of bit zero of $n_2$ of being equal to 1. The probability that the right-most bit of the sum $n_1 + n_2$ of being equal to 1 is $\frac{1}{2}\alpha_0 + \frac{1}{2}(1 - \alpha_0) = \frac{1}{2}$. Therefore, bit zero is distributed $Bern(0.5)$.

For bits numbered from 1 to $d-1$, we must also consider the binary carry-in. Let $0 \leq \alpha_i \leq 1$, $i \in \{1, ..., d-1\}$ be the probability of the right-most bit of the sum of the $i^{th}$ binary carry-in plus the $i^{th}$ bit of $n_2$ of being equal to 1. Therefore, the probability that this right-most bit plus the $i^{th}$ bit of $n_1$ of being equal to 1 is again $\frac{1}{2}\alpha_i + \frac{1}{2}(1-\alpha_i) = \frac{1}{2}$, $\forall i \in \{1, ..., d-1\}$. Hence, bits zero up to $d-1$ of $n_1 + n_2$ are identically distributed as $Bern(0.5)$. Finally, there only remains to consider bit $d$. If there is a carry-out at bit $d-1$, which becomes a binary carry-in for bit $d$, then it takes value 1 with an unknown probability value $\alpha_d$.

We now show that the probability distributions of bits zero up to $d-1$ are also independent. The bits of $n_1$ are i.i.d. $Bern(0.5)$ so that there are $2^d$ possible configurations, for values $0 \leq n_1 \leq 2^d - 1$, each one with probability $2^{-d}$. Adding $n_2$ to $n_1$ results in again $2^d$ equiprobable configurations, either

$$n_2 \leq n_1 + n_2 \leq 2^d - 1 \quad \text{or} \quad 2^d \leq n_1 + n_2 \leq 2^d + n_2 - 1.$$

In the first case, bit $d$ takes value 0 whereas it takes value 1 in the second case. However, observe that the range of bits zero up to $d-1$ is the same as for the bits of $n_1$, but not in the same order, that is, configurations from $n_2$ to $2^d - 1$ followed by those from 0 to $n_2 - 1$, each one with probability $2^{-d}$. Hence these right-most $d$ bits are also i.i.d. $Bern(0.5)$ whereas the probability distribution of bit $d$ is unknown, as stated above. □

Using IEEE Standard 754 floating-point (Stallings, 2009), real numbers on computers are represented with three components: the binary sign-bit $S$, the exponent $E$-field, and the mantissa $M$ or significand, the length of it determines the precision to which numbers can be represented. In single (32-bit) precision, a floating-point real number $F$ is given by

$$F = (-1)^S \times \begin{cases} 2^{-126} \times (\frac{M}{2^{23}}) & \text{if} \quad E = 0 \\ 2^{E-127} \times (1 + \frac{M}{2^{23}}) & \text{if} \quad 1 \leq E \leq 254 \\ \infty & \text{if} \quad E = 255, M = 0 \\ NaN & \text{if} \quad E = 255, M > 0 \end{cases}$$

where value 127 ($=2^7 - 1$) is called the bias. The 23-bit mantissa $M$ takes values from 0 to $2^{23} - 1$. Value 1 in $(1 + M2^{-23})$ when $1 \leq E \leq 254$ is called the hidden bit as it is not explicitly represented. The $E$-field with 8 bits ($0 \leq E \leq 2^8 - 1 = 255$) can represent both positive and negative exponents. An exponent of zero means that 127 is stored in the $E$-field. A stored value of $E = 90$ indicates an exponent of -37. If $E = 0$ and $M = 0$, than $F = \pm 0$, the signed zeros. Plus infinity and minus infinity are obtained with $E = 255$ and $M = 0$ and the appropriate sign-bit $S$. Finally, when $E = 255$ and $M > 0$, the $NaN$ (*Not a Number*) does not represent a real number.

Consider now the multiplication of two non-zero floating-point numbers $F_1$ and $F_2$. This type of operation appears in the computation of non-zero components of the scalar product of two vectors. For practical reasons, assume that both numbers $F_1$ and $F_2$ are of a reasonable size, that is, neither too small nor too large. To prevent under- and over-flow situations, we chose the following parameters:

$$F_1 = (-1)^{S_1} \times\ 2^{E_1 - 127} \times (1 + \frac{M_1}{2^{23}}), \qquad S_1 \text{ binary}, 64 \leq E_1 \leq 191, M_1 \geq 0,$$

$$F_2 = (-1)^{S_2} \times\ 2^{E_2 - 127} \times (1 + \frac{M_2}{2^{23}}), \qquad S_2 \text{ binary}, 64 \leq E_2 \leq 189, M_2 \geq 0.$$

The range of floating-point numbers $F_1$ and $F_2$ is approximately $\pm$ ($10^{-19}$ to $10^{19}$). Note that in simplex codes, numbers smaller in absolute value than $10^{-19}$ are already set at zero. Moreover, if an overflow could occur at some point within the multiplication $w^\top A_j, j \notin B$, it means that this could also occur at each iteration within the computation of the reduced cost $\bar{c}_j$. Hence, as shown bellow, the result $F$ of the multiplication $F_1 F_2$ is a finite non-zero floating-point number

$$F = (-1)^S \times\ 2^{E-127} \times (1 + \frac{M}{2^{23}}), \qquad S \text{ binary}, 1 \leq E \leq 254, M \geq 0.$$

Essentially, the multiplication of $F_1$ by $F_2$ is based on bit-additions. The sign-bit $S = S_1 + S_2$ modulo 2. The $M$-field comes from the multiplication of $(1 + \frac{M_1}{2^{23}})(1 + \frac{M_2}{2^{23}})$ that can be seen as a series of additions which results in a number truncated to 23 bits and a binary carry, denoted $E_{\text{M}}$, used in the computation of $E = E_1 + E_2 + E_{\text{M}} - 127$. Indeed, consider the (kind of generic) example of Figure 1 for the multiplication of a 23-bit mantissa $M_1$ by 1.11010.

$$
\begin{array}{cccccccc|cccccc}
 & & & & & & & 1. & d_{22} & d_{21} & \cdots & d_2 & d_1 & d_0 \\
 & & & & & & \times & 1. & 1 & 1 & 0 & 1 & 0 & \\
\hline
 & & & & & & & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\
+ & & & & & & 1 & d_{22} & d_{21} & \cdots & d_2 & d_1 & d_0 & \\
+ & & & & & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & & \\
+ & & & & 1 & d_{22} & d_{21} & \cdots & d_2 & d_1 & d_0 & & & \\
+ & & & 1 & d_{22} & d_{21} & \cdots & d_2 & d_1 & d_0 & & & & \\
+ & & 1 & d_{22} & d_{21} & \cdots & d_2 & d_1 & d_0 & \multicolumn{5}{c}{\text{– truncated area –}} \\
\hline
= & & 1. & m_{22} & m_{21} & \cdots & m_2 & m_1 & m_0 & \multicolumn{5}{l}{\text{without a carry}} \\
 & 1 & m_{23}. & m_{22} & \cdots & m_3 & m_2 & m_1 & m_0 & \multicolumn{5}{l}{\text{with a carry}} \\
 & & 1. & m_{23} & m_{22} & \cdots & m_3 & m_2 & m_1 & \multicolumn{5}{l}{\text{with a carry, but shifted}}
\end{array}
$$

Figure 1: Mantissas multiplication

First, observe the presence of the hidden bit **1.** for both mantissa fields, where $d_{22}...d_0$ are the mantissa-bits of $M_1$. The hidden bit also appears in the final result, where $m_{23}...m_0$ are the possible mantissa-bits of $M$. It appears as **1.** if there is no carry on the hidden bit of $M$ or as 1 $m_{23}$**.** if there is one. In the latter case, the number is shifted one position to the right, that is, the new number $1.m_{23}m_{22}...m_1$ must be multiplied by 2, hence a value $E_{\text{M}} = 1$ to be added to the $E$-field. Therefore, with the above assumptions on the values of $E_1, E_2$, and $E_M$, we have $64 \leq E_2 + E_{\text{M}} \leq 190$, that is, this sum can be written using the 7 right-most bits of the $E$-field, i.e., $1 \leq E_1 + E_2 + E_{\text{M}} - 127 \leq 254$. Therefore, $F = F_1F_2$ is a finite non-zero floating-point number.

**Definition 3** *A floating-point number with distribution $SEM_{32}$ is a single precision number $F$, where the sign-bit $S$, the $E$-field, and the $M$-field are independent and follow the discrete uniform distributions $S \sim U[0, 1]$, $E \sim U[64, 191]$, and $M \sim U[0, 2^{23} - 1]$.*

**Lemma 2** *Let $F_1 \sim SEM_{32}$. Except for the $8^{th}$ bit of $E_1$, all other 31 bits of $F_1$ are i.i.d. Bern(0.5).*

**Proof.** $S_1 \sim Bern(0.5)$ and from the proof of Lemma 1, the 23 bits of $M_1$ are i.i.d. $Bern(0.5)$. Given $E_1 \sim U[64, 191]$, there are 128 equiprobable configurations written on a 8-bit binary vector. These can be written from $2^6$ to $2^7 - 1$ followed by $2^7 + 0$ up to $2^7 + 2^6 - 1$. In the first half, the $8^{th}$ bit is 0 whereas it takes value 1 for the second half. Therefore, the 128 possible configurations over the 7 right-most bits are present, each one with probability $2^{-7}$. From the proof of Lemma 1, these bits are i.i.d. $Bern(0.5)$. Finally note that the $8^{th}$ bit of $E_1$ is dependent on the $7^{th}$ bit since it is equal to its binary complement. From the independence of $S_1$, $E_1$, and $M_1$, 31 bits of $F_1$ are independent, with the same $Bern(0.5)$ distribution. $\square$

We are now in position to state the properties of the product $F = F_1F_2$ of two single precision floating-point numbers.

**Lemma 3** *Assume that the two numbers $F_1$ and $F_2$ are independent. Moreover the probability distributions for the bits of $F_2$ are unknown and $F_1 \sim SEM_{32}$ Then, the product $F = F_1F_2$ is a finite non-zero floating-point number such that, except for the $8^{th}$ bit of $E$ and the $23^{th}$ bit of $M$, all remaining 30 bits of $F$ are i.i.d. Bern(0.5).*

**Proof.** Because $1 \leq E \leq 254$, $F$ is a finite non-zero number. The sign-bit $S = S_1 + S_2$ modulo 2, and one can verify that $Pr[S = 1] = \frac{1}{2}$, i.e., $S \sim Bern(0.5)$. Regarding the $E$-field, we are adding $E_2 + E_{\text{M}}$ of unknown bit distributions to $E_1$. By Lemma 2, the 7 right-most bits of $E_1$ are i.i.d. $Bern(0.5)$. By Lemmas 1, the 7 right-most bits of the sum $E_1 + (E_2 + E_{\text{M}})$ are i.i.d. $Bern(0.5)$ (the subtraction of the constant value 127 does not change the bit distributions) whereas the $8^{th}$ bit of this sum is of unknown probability distribution.

As for the $M$-field, observe that the series of additions (refer to Figure 1) can be decomposed in two parts. Firstly, add all but the last row: this is a 24-bit vector (including the hidden bit) of unknown bit distributions. Secondly, add this sum to the last row $1.d_{22}...d_0$ where $d_{22}...d_0$ are the bits of $M_1$ i.i.d. $Bern(0.5)$. By Lemma 1, $m_{22}...m_0$ are i.i.d. $Bern(0.5)$ but $m_{23}$ is of unknown distribution, being the sum of a carry to 1. Since $m_{23}$ can appear in the final result, the 22 right-most bits are i.i.d. $Bern(0.5)$ and the $23^{th}$ ($m_{22}$ if there is no carry on the hidden bit, $m_{23}$ otherwise) is of unknown distribution. The final result on the 30 bits i.i.d. $Bern(0.5)$ follows from the independence of the $S$, $E$, and $M$ fields. □

**Proposition 1** *Let $A_j, j \notin B$ be a non-basic incompatible column (i.e., $\bar{A}^z_j \neq 0$) and let $v$ be a row vector of $(m-p)$ components, each of which being a floating-point number $v_i \sim SEM_{32}$, $i \in Z$. If there is no overflow nor $NaN$ situations, then the conditional probability that $v^\top \bar{A}^z_j = 0$ is less than or equal to $2^{-30}$, that is,*

$$P[v^\top \bar{A}^z_j = 0 \mid \bar{A}^z_j \neq 0] \leq \frac{1}{2^{30}}.$$

**Proof.** Let $\bar{a}^z_{i_k j} \neq 0$ for $k \in K$, where $1 \leq |K| \leq m-p$ is the number of non-zero coefficients of $\bar{A}^z_j$. For the ease of the presentation, we do not write the conditional event $\bar{A}^z_j \neq 0$ in the probability expressions.

If $|K| = 1$, then $P[v^\top \bar{A}^z_j = 0] = P[v_{i_1}\bar{a}^z_{i_1 j} = 0 \mid v_{i_1} \neq 0, \bar{a}^z_{i_1 j} \neq 0] = 0$. If $|K| \geq 2$, then

$$P[v^\top \bar{A}^z_j = 0] = P[\sum_{k\in K} v_{i_k}\bar{a}^z_{i_k j} = 0] = P[v_{i_1}\bar{a}^z_{i_1 j} = -\sum_{k\geq 2} v_{i_k}\bar{a}^z_{i_k j}].$$

Let $W = -\sum_{k\geq 2} v_{i_k}\bar{a}_{i_k j}$. Define $\Omega$ to be the set of all possible 32-bit values for $W$ and assume that $\omega \in \Omega$ is a finite floating-point number. Hence,

$$P[v^\top \bar{A}^z_j = 0] = P[v_{i_1}\bar{a}^z_{i_1 j} = W] = \sum_{\omega\in\Omega} P[W = \omega] \times P[v_{i_1}\bar{a}^z_{i_1 j} = \omega].$$

Since $v_{i_1}\bar{a}^z_{i_1 j} \neq \pm 0$, define $\Omega_0 = \Omega \setminus \{\pm 0\}$. From Lemma 3, 30 bits of $v_{i_1}\bar{a}^z_{i_1 j}$ are i.i.d. $Bern(0.5)$ while the two other bit-distributions are unknown. From probability theory, recall that for two events $\mathcal{E}_1$ and $\mathcal{E}_2$, $P[\mathcal{E}_1 \cap \mathcal{E}_2] \leq P[\mathcal{E}_i], i \in \{1,2\}$. Hence, the probability that $P[v_{i_1}\bar{a}^z_{i_1 j} = \omega]$, $\omega \in \Omega_0$ is the probability of being identical on each of the 32 bits which is less than or equal to the probability of being identical on the 30 bits i.i.d. $Bern(0.5)$. Therefore $P[v_{i_1}\bar{a}^z_{i_1 j} = \omega] \leq 2^{-30}$, $\omega \in \Omega_0$. Consequently

$$P[v^\top \bar{A}^z_j = 0] = \sum_{\omega\in\Omega_0} P[W = \omega] \times P[v_{i_1}\bar{a}^z_{i_1 j} = \omega] \leq 2^{-30} \sum_{\omega\in\Omega_0} P[W = \omega] \leq 2^{-30}.$$

In conclusion, if a non-basic variable is incompatible, it might be wrongly identified as compatible with a probability error less than or equal to $2^{-30}$. □

Similar results can be derived for a 64-bit computer words. A floating-point number with distribution $SEM_{64}$ is a double precision number $F$, where the sign-bit $S$, the 11-bit $E$-field, and the 52-bit $M$-field are independent and follow the discrete uniform distributions $S \sim U[0,1]$, $E \sim U[512, 1535]$, and $M \sim U[0, 2^{52}-1]$.

**Proposition 2** *Let $A_j, j \notin B$ be a non-basic incompatible column (i.e., $\bar{A}^z_j \neq 0$) and let $v$ be a row vector of $(m-p)$ components, each of which being a floating-point number $v_i \sim SEM_{64}, i \in Z$. If there is no overflow nor $NaN$ situations, then $P[v^\top \bar{A}^z_j = 0 \mid \bar{A}^z_j \neq 0] \leq 2^{-62}$.*

## 5 A Two-dimensional Reduced Cost Criterion for the Primal Simplex

There are several ways to integrate the positive edge criterion in a primal simplex algorithm. One is presented in the following paragraphs. It uses two dimensions; one for the usual reduced cost, the other to indicate if a variable is compatible or not. The first computes $\bar{c}_j = c_j - \pi^\top A_j$, $j \notin B$, where $\pi$ is the vector of dual variables for LP whereas the second needs a similar computation, that of $w^\top A_j, j \notin B$. The positive edge identifies $\mathrm{C}_w = \{j \notin B | w^\top A_j = 0\}$, the index set of non-basic compatible variables with respect to $v^\top Q^{\mathrm{Z}}$ and the index set of incompatible ones, i.e., those in $\mathrm{I}_w = \{j \notin B | w^\top A_j \neq 0\}$.

Let $\bar{c}_{j_{min}}$ be the smallest reduced cost for a non-basic variable indexed by $j_{min} \notin B$ and let $\bar{c}_{j_w}$ be the smallest reduced cost for a non-basic variable compatible with respect to $v^\top Q^{\mathrm{Z}}$ and indexed by $j_w \in \mathrm{C}_w$. Observe that $j_{min} \in \mathrm{C}_w \cup \mathrm{I}_w$. The stopping rule for optimality remains the same, that is, the current solution is optimal if $\bar{c}_{j_{min}} \geq 0$. Hence initialize both $\bar{c}_{j_{min}}$ and $\bar{c}_{j_w}$ at zero. Therefore we have the following relations: $\bar{c}_{j_{min}} \leq \bar{c}_{j_w} \leq 0$.

With the positive edge criterion, compatible variables with negative reduced cost should be preferred to enter the basis such that non-degenerate pivots are executed, yielding a strict improvement of the objective function value. This is done in priority except if $\bar{c}_{j_{min}}$ is much smaller than $\bar{c}_{j_w}$. This can be controlled with the threshold parameter $\psi > 1$, e.g., $\psi = 5$. Hence the selection rule becomes:

$$\text{if} \quad \bar{c}_{j_{min}} < 0 \quad \text{and} \quad \psi \bar{c}_{j_w} < \bar{c}_{j_{min}}, \quad \text{then select} \quad x_{j_w} \quad \text{else} \quad x_{j_{min}}.$$

Note that as long as compatible variables enter into the basis, the status of a variable of being in $\mathrm{C}_w$ or not remains unchanged. One can benefit from that by keeping track of the compatible variables, updating their reduced costs, and verifying if one has a negative value. Naturally, the above rule can be used in the context of partial pricing. It can also be combined with the steepest edge criterion to compute the value of the step size for compatible variables.

Finally, the consequence of selecting an incompatible variable identified as compatible (which has only a one-in-a-billion chance of occurring) is the same as in the context of the primal simplex algorithm, that is, a degenerate pivot.

## 6 An External Algorithm

Computational experiments were made with CPLEX (`http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`). Since we do not have access to the source code of IBM ILOG to fully integrate the positive edge criterion in the primal simplex algorithm of CPLEX, this section proposes a simple implementation called the External Algorithm to evaluate the potential of this new criterion. A series of partial problems, each composed of a subset of the original variables, are used for solving LP. Two external procedures are utilized: the first identifies non-basic compatible variables with respect to $v^\top Q^{\mathrm{Z}}$ while the second computes and identifies negative reduced cost variables.

The External algorithm is stated as Algorithm 6.1. The algorithm starts at Step 0 with a feasible basis $A_B$. If the solution is non degenerate, then it directly solves LP at Step 1 and stops. Otherwise, it makes the partition of the rows of LP and of the inverse $Q$ of the basis according to sets P and Z at Step 3. A phase I can be used to identify such an initial basis, or as shown in Elhallaoui et al. (2007), the partition of $Q$ can be done if a solution with $p < m$ non-degenerate variables is known, e.g., from a heuristic solution. At Step 4, a random vector $v$ is selected and vector $w^\top = v^\top Q^{\mathrm{Z}} \in \mathbb{R}^m$ is computed. At Step 5, an external procedure uses the positive edge criterion to determine $\mathrm{C}_w$ and $\mathrm{I}_w$.

The External algorithm does not work with the (fully) reduced problem RP. Actually, the *partial problem* $\mathrm{LP}_{\mathrm{C}_w \bar{\mathrm{I}}_w}$ defined in (3) contains only the current basic variables $x_{\mathrm{B}}$ and the non-basic compatible variables

$x_{\text{C}_w}$ but it considers all rows of $A$, keeping up to date the $m$-dimensional vector $\pi$ of the dual variables of LP:

$$\begin{aligned} \text{minimize} \quad & c_{\text{B}}^\top x_{\text{B}} + c_{\text{C}_w}^\top x_{\text{C}_w} \\ \text{subject to:} \quad & A_{\text{B}} x_{\text{B}} + A_{\text{C}_w} x_{\text{C}_w} = b, \quad x_{\text{B}}, x_{\text{C}_w} \geq 0. \end{aligned} \tag{3}$$

Indeed, the use of the reduced problem RP would need to add a third external procedure for computing the dual variables associated with the eliminated rows. Working with $\text{LP}_{\text{C}_w\bar{\text{I}}_w}$, the reduced costs of the incompatible variables are easier to obtain at the price of a larger basis and a higher computing time per iteration.

At Step 6, construct the partial problem $\text{LP}_{\text{C}_w\bar{\text{I}}_w}$ and solve it at Step 7 until optimality is reached or $\lceil \sigma m \rceil$ pivots are performed $(0 < \sigma < 1)$. At Step 8, externally compute the reduced cost of all non-basic incompatible variables $x_j, j \in \text{I}_w$ using the vector $\pi$ of the dual variables of $\text{LP}_{\text{C}_w\bar{\text{I}}_w}$ and let $\bar{\text{I}}_w$=$\{j \in \text{I}_w | \bar{c}_j < 0\}$ be the index set of the negative reduced cost variables.

Optimality is tested at Step 9: if $\bar{\text{I}}_w = \emptyset$ and the partial problem is optimal, then no variables have a negative reduced cost, and the current solution of $\text{LP}_{\text{C}_w\bar{\text{I}}_w}$ is also optimal for LP. If $\bar{\text{I}}_w = \emptyset$ but $\text{LP}_{\text{C}_w\bar{\text{I}}_w}$ is not optimal, return to Step 7 to carry on the process of solving the partial problem.

At Step 10, $\bar{\text{I}}_w \neq \emptyset$. Add all these $|\bar{\text{I}}_w|$ negative reduced cost incompatible variables to the partial problem to form $\text{LP}_{\text{C}_w\bar{\text{I}}_w}$, an augmented partial problem (4), stated as follows:

$$\begin{aligned} \text{minimize} \quad & c_{\text{B}}^\top x_{\text{B}} + c_{\text{C}_w}^\top x_{\text{C}_w} + c_{\bar{\text{I}}_w}^\top x_{\bar{\text{I}}_w} \\ \text{subject to:} \quad & A_{\text{B}} x_{\text{B}} + A_{\text{C}_w} x_{\text{C}_w} + A_{\bar{\text{I}}_w} x_{\bar{\text{I}}_w} = b, \;\; x_{\text{B}}, x_{\text{C}_w}, x_{\bar{\text{I}}_w} \geq 0. \end{aligned} \tag{4}$$

At the final step, the augmented problem $\text{LP}_{\text{C}_w\bar{\text{I}}_w}$ is solved to optimality or until $\lceil \delta m \rceil$ pivots are performed $(0 < \delta < 1)$. Then the method identifies a new basis by returning to Step 0.

**Algorithm 6.1 External Algorithm**

```
Step 0.  Identify a basis $A_B$ for LP with $p \leq m$ non-degenerate basic variables. Let $Q = A_B^{-1}$.
Step 1.  If $p = m$, then solve LP and stop.
Step 2.  Otherwise, let $\bar{b} = Qb$, $\text{P} = \{i \in \{1, ..., m\} \mid \bar{b}_i > 0\}$ and $\text{Z} = \{i \in \{1, ..., m\} \mid \bar{b}_i = 0\}$.
Step 3.  Partition LP and $Q$ according to row-sets P and Z.
Step 4.  Select a random vector $v$ and compute the $m$-vector $w^\top = v^\top Q^{\text{Z}}$.
Step 5.  Let $\text{C}_w$ [$\text{I}_w$] be the set of non-basic compatible [incompatible] variables with respect to $v^\top Q^{\text{Z}}$.
Step 6.  Construct $\text{LP}_{\text{C}_w\bar{\text{I}}_w}$ by removing from LP the non-basic incompatible variables in $\text{I}_w$.
Step 7.  Solve $\text{LP}_{\text{C}_w\bar{\text{I}}_w}$ until $\lceil \sigma m \rceil$ pivots are performed or optimality is reached.
Step 8.  Let $\bar{\text{I}}_w = \{j \in \text{I}_w | \bar{c}_j < 0\}$ be the index set of the negative reduced cost incompatible variables.
Step 9.  If $\bar{\text{I}}_w = \emptyset$, then stop if $\text{LP}_{\text{C}_w\bar{\text{I}}_w}$ is optimal or go to Step 7 if $\text{LP}_{\text{C}_w\bar{\text{I}}_w}$ is not optimal.
Step 10. Otherwise, construct $\text{LP}_{\text{C}_w\bar{\text{I}}_w}$ by appending the $|\bar{\text{I}}_w|$ negative reduced cost variables to $\text{LP}_{\text{C}_w\bar{\text{I}}_w}$.
Step 11. Solve $\text{LP}_{\text{C}_w\bar{\text{I}}_w}$ until $\lceil \delta m \rceil$ pivots are performed or optimality is reached. Go to Step 0.
```

If we could have access to the information of the $LU$-factorization of the basis (Bazaraa, Jarvis, and Sherali, 2010), it would be possible to work with the reduced problem RP that contains only $p$ rows. Having the inverse $L^{-1}$ and the row-permutation matrix of this factorization, it would be possible to completely reduce the problem since we would know the row associations and the permutations to do on the $A$ and $Q$ matrices, which means that we could solve RP. Then we could compute the dual variables by adding the eliminated rows and the basic variables associated with each of these rows. In Section 7, we present the results of the External algorithm (denoted External-$\text{LP}_{\text{C}_w}$ in the Tables) and we also provide the computational results of the so-called External-RP algorithm, that is, of what we could obtain if we knew the $LU$-factorization information. To estimate the time of the External-RP algorithm, we construct and solve the first reduced problem RP encountered and we compare its solution time to that of the first partial problem $\text{LP}_{\text{C}_w\bar{\text{I}}_w}$. Then, we extrapolate the total time of the External-RP algorithm by using this time ratio.

## 7 Computational Experiments

This section presents computational experiments conducted with the External algorithm. We begin with medium-sized aircraft *fleet assignment* (FA) instances, followed by large-scale *manpower planning problems* (MPP). We complete the tests with some *Mittelmann*'s instances used for the benchmarking of commercial LP solvers.

Tests were performed on a 2.8GHz PC running Linux and comparisons are made with CPLEX 12.1 with default settings (hybrid reduced-cost and Devex pricing). The External algorithm uses $SEM_{32}$ random variables and two parameters, $\sigma$ and $\delta$, that are related to the problem size. More precisely, $\sigma = 0.2$ if $m \leq 10\ 000$, $\sigma = 0.1$ if $10\ 000 < m \leq 25\ 000$, $\sigma = 0.05$ if $25\ 000 < m \leq 75\ 000$, $\sigma = 0.02$ if $75\ 000 < m \leq 400\ 000$, and $\sigma = 0.01$ if $m > 400\ 000$. The value of $\delta$ is set to $2\sigma$.

The algorithm starts with an initial basis obtained from a phase I computed by CPLEX. In all tables of results, the columns denoted *pivots* represent the number of pivots executed by the primal simplex algorithm of CPLEX while the column *loops* represents the number of loops executed by the External algorithm, i.e., the number of times the inverse basis $Q$ is identified at Step 0. The column *factor* corresponds to the reduction factor of the total computational time (i.e., the ratio of the primal CPLEX CPU time to the time estimation provided by the External-RP algorithm). All computational times are given in seconds. Finally note that the number of pivots and the number of loops are identical for both External-LP$_{C_w}$ and External-RP algorithms.

### 7.1 FA Instances

Airline fleet assignment instances FA consist in maximizing the profits of assigning an aircraft type to each flight segment over a one-week planning horizon. The routing must respect maintenance requirements and aircraft type availability. The instances are generated from real data with 2505 flight segments and four types of aircraft. The variables are flight sequences between maintenance bases. These problems have a set partitioning constraint per flight segment, an availability constraint per aircraft type, and a flow conservation constraint between flight sequences at maintenance bases. Some variables are also bounded from above. These problems are typical master problem instances that need to be solved at each iteration of a branch-and-price algorithm. Additionally, these instances were used in Raymond, Soumis, and Orban (2010). Table 1 gives the number of constraints and variables for these medium-sized instances along with the average proportion of degenerate variables encountered in the solution process with IPS (Raymond, Soumis, and Orban, 2010). Note that the degeneracy difficulties depend more on the path taken by an algorithm rather than on the optimal values of the variables. The number of constraints is approximatively 5000, the number of variables close to 25 000, and the average number of degenerate variables about 65%.

Table 1: Characteristics of the FA instances

| instance | constraints | variables | degeneracy | instance | constraints | variables | degeneracy |
|---|---|---|---|---|---|---|---|
| FA 6 | 5067 | 17 594 | 68% | FA13 | 5159 | 25 746 | 65% |
| FA 7 | 5159 | 20 434 | 59% | FA14 | 5159 | 22 641 | 71% |
| FA 8 | 5159 | 21 437 | 65% | FA15 | 5182 | 23 650 | 63% |
| FA 9 | 5159 | 23 258 | 66% | FA16 | 5182 | 23 990 | 64% |
| FA10 | 5159 | 24 492 | 66% | FA17 | 5182 | 24 282 | 65% |
| FA11 | 5159 | 24 812 | 66% | FA18 | 5182 | 24 517 | 65% |
| FA12 | 5159 | 24 645 | 66% | FA19 | 5182 | 24 875 | 65% |

Table 2 presents the results for the FA instances. On average, the Primal Simplex of CPLEX finds an optimal solution in 473 seconds compared to only 72 seconds for the External algorithm executing nine loops. The estimation given by the External-RP algorithm is slightly faster at 69 seconds and, on average, it is more than 7 times faster than CPLEX. Moreover, the number of pivots of the simplex algorithm is almost two times greater with CPLEX compared to the simple implementation. It should be emphasized that the CPU savings become very significant when thousands of such problems need to be solved during a branch-and-price algorithm.

Table 2: Computational results for the FA instances

| | CPLEX PRIMAL | | EXTERNAL-LP$_{C_w}$ ALGORITHM | | | EXTERNAL-RP ALGORITHM | |
|---|---|---|---|---|---|---|---|
| instance | pivots | time | loops | pivots | time | time | factor |
| FA 6 | 63 409 | 257 | 8 | 28 204 | 36 | 35 | 7.34 |
| FA 7 | 65 689 | 289 | 8 | 28 589 | 40 | 39 | 4.90 |
| FA 8 | 61 180 | 260 | 8 | 43 607 | 60 | 59 | 4.19 |
| FA 9 | 81 806 | 387 | 10 | 43 440 | 63 | 62 | 4.35 |
| FA10 | 97 234 | 528 | 8 | 57 138 | 91 | 89 | 5.03 |
| FA11 | 103 465 | 538 | 8 | 72 672 | 110 | 105 | 9.28 |
| FA12 | 91 175 | 471 | 9 | 37 955 | 60 | 58 | 5.35 |
| FA13 | 100 014 | 561 | 10 | 66 763 | 102 | 88 | 7.79 |
| FA14 | 114 632 | 584 | 12 | 52 771 | 76 | 72 | 5.78 |
| FA15 | 103 076 | 528 | 11 | 73 004 | 104 | 101 | 9.26 |
| FA16 | 82 797 | 420 | 8 | 43 687 | 59 | 57 | 5.38 |
| FA17 | 122 634 | 670 | 8 | 46 401 | 80 | 78 | 9.44 |
| FA18 | 121 050 | 671 | 9 | 53 818 | 74 | 71 | 11.77 |
| FA19 | 87 201 | 469 | 8 | 42 516 | 58 | 57 | 13.40 |
| AVG | 92 525 | 473 | 9 | 49 326 | 72 | 69 | 7.38 |

## 7.2 MPP Instances

Manpower planning problems (MPP) are two instances provided by the Montreal based company AD OPT, a division of KRONOS. These instances have approximately 99 000 constraints and 450 000 variables. The average proportion of degenerate variables in the basis at Step 7 of the solution of the partial problem LP$_{C_w \bar{I}_w}$ in the External algorithm is 80% in both cases.

Two tables are shown. On the one hand, Table 3 presents the results when an initial basis is already available, as in a column generation scheme for solving a series of master problems. A Phase I, solved by CPLEX, is used to construct this initial basis but its CPU time and its number of pivots are not taken into account in Table 3. On the other hand, the CPU time and the number of pivots involved in the solution of the Phase I are added to the External algorithm results in Table 4 whereas the number of loops remains the same. This second situation is kind of worst case: the computing time of the External algorithm includes Phase I time while CPLEX starts its solution process with its well-tuned *presolve*.

Table 3: Results for the MPP instances – Initial basis available

| | CPLEX PRIMAL | | EXTERNAL-LP$_{C_w}$ ALGORITHM | | | EXTERNAL-RP ALGORITHM | |
|---|---|---|---|---|---|---|---|
| instance | pivots | time | loops | pivots | time | time | factor |
| MPP1 | 725 200 | 2337 | 188 | 193 614 | 432 | 365 | 6.40 |
| MPP2 | 734 438 | 1884 | 158 | 148 144 | 372 | 321 | 5.87 |
| AVG | 729 819 | 2110 | 173 | 170 879 | 402 | 343 | 6.14 |

When an initial basis is available (Table 3), CPLEX needs on average 2110 seconds to find an optimal solution whereas the External algorithm requires 402 seconds with 173 loops. The External-RP algorithm is 15% faster at 343 seconds and, on average, it is more than 6 times faster than CPLEX. Note that the External algorithm decreases the number of pivots from 729 819 pivots for CPLEX to only 170 879.

When no initial basis is available (Table 4), CPLEX is much faster as its *presolve* procedure followed by the Primal Simplex only needs 537 seconds on average compared to the previous 2100 seconds displayed in Table 3.

Table 4: Results for the MPP instances – No initial basis available

| | CPLEX PRIMAL | | EXTERNAL-$\text{LP}_{C_w}$ ALGORITHM | | | EXTERNAL-RP ALGORITHM | |
|---|---|---|---|---|---|---|---|
| instance | pivots | time | loops | pivots | time | time | factor |
| MPP1 | 437 453 | 540 | 188 | 194 975 | 443 | 375 | 1.44 |
| MPP2 | 387 871 | 533 | 158 | 149 253 | 385 | 332 | 1.61 |
| AVG | 412 662 | 537 | 173 | 172 114 | 414 | 353 | 1.52 |

However, the External algorithm (including the Phase I CPU time) is still faster at 414 seconds, estimated at 353 seconds by the External-RP algorithm. In this worst case scenario, our simple implementation is therefore 52% faster than CPLEX. Observe also that the proposed algorithm decreases the number of pivots by a factor of 2.39, that is, on average, from 412 662 pivots for CPLEX versus 172 114 for the External algorithm. Therefore, even though both algorithms start from scratch, the positive edge criterion is very efficient in the reduction of the number of pivots, an illustration of its potential.

## 7.3 Mittelmann's Instances

The instances of this section are available at http://plato.asu.edu/ftp/lpcom.html and are benchmark instances for commercial linear programming solvers. We chose these large-scale problems because they are highly degenerate. Table 5 gives the number of constraints and variables of each instance along with the average proportion of degenerate variables in the basis at Step 7 of the solution of the partial problem $\text{LP}_{C_w \bar{I}_w}$ in the External algorithm. One can observe that there are two types of instances: the proportion of degenerate variables for the PDS instances is about 83% while it is close to 50% for Fome12 and Fome13.

Table 5: Characteristics of Mittelmann's instances

| instance | constraints | variables | degeneracy | instance | constraints | variables | degeneracy |
|---|---|---|---|---|---|---|---|
| PDS-20 | 33 874 | 108 175 | 82% | PDS-80 | 129 181 | 434 580 | 84% |
| PDS-30 | 49 944 | 158 489 | 82% | PDS-90 | 142 823 | 475 448 | 84% |
| PDS-40 | 66 844 | 217 531 | 81% | PDS-100 | 156 243 | 514 577 | 84% |
| PDS-50 | 83 060 | 274 814 | 82% | | | | |
| PDS-60 | 99 431 | 336 421 | 83% | Fome12 | 24 285 | 48 920 | 54% |
| PDS-70 | 114 944 | 390 005 | 83% | Fome13 | 48 569 | 97 840 | 46% |

The computational results are presented in Table 6 for the PDS instances and in Table 7 for the two Fome problems. At the start, no initial basis is available for these problems. This means that CPLEX uses its *presolve* procedure whereas in the External algorithm the Phase I CPU time and its additional number of pivots are included in the presented results.

Regarding the computational time, the *presolve* procedure followed by the Primal Simplex of CPLEX is slightly faster on the two smallest PDS problems (in CPU time, 10 and 27 versus 12 and 32 seconds). However, on average for the nine instances, it needs 631 seconds compared to the External algorithm requiring 451 seconds with 49 loops, estimated at 361 seconds by the External-RP algorithm. The simple implementation is 55% faster than CPLEX. Observe again that the proposed algorithm decreases the number of pivots by a factor of 3.57, that is, on average, from 515 676 pivots for CPLEX versus 144 521 for the External algorithm.

For the two Fome instances of Table 7, the External algorithm did not perform well. CPLEX is about ten times faster, the number of loops is quite large on both Fome12 and Fome13 (212 and 423 loops, respectively), hence the total number of pivots.

Table 6: Results for Mittelmann's PDS instances – No initial basis available

| | CPLEX PRIMAL | | EXTERNAL-LP$_{C_w}$ ALGORITHM | | | EXTERNAL-RP ALGORITHM | |
|---|---|---|---|---|---|---|---|
| instance | pivots | time | loops | pivots | time | time | factor |
| PDS-20 | 51 419 | 10 | 18 | 15 420 | 14 | 12 | 0.83 |
| PDS-30 | 106 854 | 27 | 21 | 30 365 | 39 | 32 | 0.84 |
| PDS-40 | 281 419 | 135 | 35 | 73 294 | 121 | 95 | 1.42 |
| PDS-50 | 517 983 | 406 | 48 | 135 094 | 267 | 212 | 1.92 |
| PDS-60 | 614 014 | 631 | 59 | 185 970 | 402 | 324 | 1.95 |
| PDS-70 | 708 011 | 859 | 60 | 224 745 | 564 | 447 | 1.92 |
| PDS-80 | 756 449 | 1060 | 64 | 288 567 | 788 | 630 | 1.68 |
| PDS-90 | 803 204 | 1350 | 70 | 329 942 | 911 | 727 | 1.86 |
| PDS-100 | 801 739 | 1202 | 64 | 317 294 | 952 | 773 | 1.55 |
| AVG | 515 676 | 631 | 49 | 144 521 | 451 | 361 | 1.55 |

Table 7: Results for Mittelmann's Fome instances – No initial basis available

| | CPLEX PRIMAL | | EXTERNAL-LP$_{C_w}$ ALGORITHM | | | EXTERNAL-RP ALGORITHM | |
|---|---|---|---|---|---|---|---|
| instance | pivots | time | loops | pivots | time | time | factor |
| Fome12 | 135 604 | 361 | 212 | 954 216 | 2 342 | 2 239 | 0.16 |
| Fome13 | 267 605 | 1214 | 423 | 2 054 351 | 13 370 | 12 900 | 0.09 |
| AVG | 201 605 | 788 | 318 | 1 504 284 | 7876 | 7570 | 0.13 |

## 7.4 Limitations of the External Algorithm and Possible Improvements

As mentioned previously, the External algorithm used to show the potential of the positive edge criterion is a simple implementation. On the Fome12 and Fome13, and the two smallest PDS instances, it did not improve on the performance time of CPLEX. We now investigate its behavior and propose ways to improve upon the simple implementation.

From Table 5, we observe that the percentage of degeneracy of Fome12 (54%) and Fome13 (46%) is less than the average values for FA (65%), MPP (80%), and PDS (83%) instances. Indeed, considering the reduced problem RP, there is a big difference on the size of the $p \times p$ basis if the percentage of degenerate variables is 50% rather than 80%. For the same number $m$ of constraints, the number of entries in the reduced basis is $(0.5m)^2$ in the first case whereas it is $(0.2m)^2$ in the second, that is, $0.25m^2$ compared to $0.04m^2$, a factor of 6.25 larger. As the positive edge criterion is essentially appropriate for highly degenerate problems, a part of the poor results can be explained by that.

Table 8 provides two additional informations: the density $\lambda_A$ of the constraint matrix $A$ and the average proportion $|C_w|/n$ of non-basic compatible variables in LP$_{C_w \bar{I}_w}$ compared to the total number of variables in LP for MPP, PDS and Fome instances. The instances for which the External algorithm did not perform better than CPLEX appear in boldface characters. We observe that either the density $\lambda_A$ or the relative size to LP of the partial problem LP$_{C_w \bar{I}_w}$ is significantly higher in comparison to the other instances.

Higher density can largely increase the number of pivots required to solve a linear program. This can be seen from Table 9 where we analyze the behavior of the Primal Simplex algorithm of CPLEX. In this Table, we can compare PDS-20 to Fome12 and PDS-30 to Fome13. Consider PDS-20 and Fome12, a similar analysis can be done for the other pair. Although PDS-20 is larger in size compared to Fome12 in terms of the number of constraints (33 874 vs. 24 285) and variables (108 175 vs. 48 920) and, moreover, is much more degenerate (82% vs. 54%), the solution time for Fome12 (361 seconds) is 36.1 larger than for PDS-20 (10 seconds). The

Table 8: Density $\lambda_A$ of the matrix $A$ and proportion $|C_w|/n$ of compatible variables in $\mathrm{LP}_{\mathrm{C}_w \bar{\mathrm{I}}_w}$

| instance | density $\lambda_A$ | $\|C_w\|/n$ (%) | instance | density $\lambda_A$ | $\|C_w\|/n$ (%) |
|---|---|---|---|---|---|
| MPP1 | 2.6e-5 | 26% | PDS-70 | 1.9e-5 | 31% |
| MPP2 | 2.6e-5 | 27% | PDS-80 | 1.7e-5 | 31% |
| **PDS-20** | **6.3e-5** | **32%** | PDS-90 | 1.5e-5 | 31% |
| **PDS-30** | **4.3e-5** | **32%** | PDS-100 | 1.4e-5 | 31% |
| PDS-40 | 3.2e-5 | 32% | | | |
| PDS-50 | 2.6e-5 | 32% | **Fome12** | **11.0e-5** | **55%** |
| PDS-60 | 2.1e-5 | 31% | **Fome13** | **6.0e-5** | **55%** |

increase of the number of pivots (51 419 vs. 135 604) by a factor 2.64 is not due to degeneracy (which should have the inverse effect in that case) but to the density of the columns of $A$: 11.0e-5 for Fome12 compared to 6.3e-5 for PDS-20. Degeneracy is not the main issue, density of $A$ is.

Table 9: Comparisons with CPLEX of PDS-20 and PDS-30 to Fome12 and Fome13

| instance | constraints | variables | degeneracy | CPLEX time | time ratio | density $\lambda_A$ | CPLEX pivots | pivot ratio |
|---|---|---|---|---|---|---|---|---|
| PDS-20 | 33 874 | 108 175 | 82% | 10 | | 6.3e-5 | 51 419 | |
| Fome12 | 24 285 | 48 920 | 54% | 361 | 36.10 | 11.0e-5 | 135 604 | 2.64 |
| PDS-30 | 49 944 | 158 489 | 82% | 27 | | 4.3e-5 | 106 854 | |
| Fome13 | 48 569 | 97 840 | 46% | 1214 | 71.41 | 6.0e-5 | 267 605 | 2.50 |

It is also well known that algorithms that use an oracle to add variables to a master problem need more iterations when the density of the constraint matrix is high. An example of that is the tailing-off effect in column generation approaches, see Jones et al. (1993); Lübbecke and Desrosiers (2005), and Oukil et al. (2006). Entering in the basis a high density column-vector may result in a small ratio-test with a higher probability, hence a small improvement of the objective function value. Finally observe also that for the Fome instances, 55% of the variables are compatible compared to about 30% for the other instances. This means that in each loop (212 for Fome12 and 423 for Fome13), both problems $\mathrm{LP}_{\mathrm{C}_w \bar{\mathrm{I}}_w}$ and $\mathrm{LP}_{\mathrm{C}_w \bar{\mathrm{I}}_w}$ are solved with more than half of the total number $n$ of variables, both being each time difficult to solve because of the high density of the constraint matrix.

Improvements could be done to the External algorithm, the most important one being a full integration of the positive edge criterion within a primal simplex code. This would eliminate the two external procedures and the tailing-off effect. An integration would also take advantage of partial pricing and some other strategies based on the accumulation of various informations during the solution process. Finally, remember that the positive edge identifies variables with positive step size but does not provide its value. A combination of it with the steepest edge could increase its performance, especially for dense instances.

## 8 Conclusion

In this paper, we propose a new pricing criterion for the primal simplex algorithm: the *positive edge* identifies, with a probability error less than or equal to $2^{-30}$ in single precision binary floating-point format and $2^{-62}$ in double precision format,, variables allowing for non-degenerate pivots. Its computational complexity is $O(m)$ per variable, the same as for the computation of the reduced cost.

The preliminary computational results show the high potential of this new pricing rule. Comparisons are made with the primal simplex algorithm of CPLEX. We designed a very simple algorithm that uses two external procedures for the selection of the variables that are send to CPLEX: the first identifies variables that allow for non-degenerate pivots, the second computes negative reduced costs. It has been tested on fourteen medium-sized aircraft fleet assignment instances, two large-scale manpower planning problems, and nine PDS instances from the Mittelmann library. All these problems are highly degenerate. On the first group of instances, our simple implementation is 7 times faster than CPLEX on average and the number of pivots is almost reduced by a factor 2. On the second and third groups, it is more than 50% faster and the number of pivots is decreased by 2.39 and 3.57, respectively.

It has also been tested on two additional problems, Fome12 and Fome13, from the Mittelmann library. For these highly dense problems, our simple implementation failed. The integration of the positive edge criterion within a primal simplex code should prevent such situations by eliminating the two external procedures, by working directly with the reduced problem, and by also taking advantage of strategies such as the partial pricing. Since the positive edge does not compute the value of the step size, a combination of it with the steepest edge should increase the performance of this new pricing rule, especially for highly degenerate and dense instances.

## References

Bland, R. G. 1977. New Finite Pivoting Rules for the Simplex Method. *Mathematics of Operations Research*, **2**(2), 103–107.

Charnes, A. 1952. Optimality and Degeneracy in Linear Programming. *Econometrica*, **20**, 160–170.

Bazaraa, M. S., J. J. Jarvis, H. D. Sherali. 2010. *Linear Programming and Network Flows.* Wiley.

Dantzig, G. B. 1963. *Linear Programming and Extensions.* Princeton University Press, Princeton, NJ.

Dantzig, G. B. 1949. *Programming in a Linear Structure.* Econometrica **17**, 73–74.

Elhallaoui, I., D. Villeneuve, F. Soumis, and G. Desaulniers. 2005. Dynamic Aggregation of Set Partitioning Constraints in Column Generation. *Operations Research* **53**(4), 632–645.

Elhallaoui, I., A. Metrane, G. Desaulniers, and F. Soumis. 2007. An Improved Primal Simplex Algorithm for Degenerate Linear Programs. Forthcoming: *INFORMS Journal on Computing.*

Elhallaoui, I., A. Metrane, G. Desaulniers, and F. Soumis. 2010. Multi-phase Dynamic Constraint Aggregation for Set Partitioning Type Problems. *Mathematical Programming* **123**(2), 345–370.

Forrest, J.J., and D. Goldfarb. 1992. Steepest-Edge Simplex Algorithms for Linear Programming. *Mathematical Programming* **57**(3), 341–374.

Fukuda, K. 1982. *Oriented Matroid Programming.* Ph.D. Dissertation, University of Waterloo, Canada.

Haase, K., G. Desaulniers, and J. Desrosiers. 2001. Simultaneous Vehicle and Crew Scheduling in Urban Mass Transit Systems. *Transportation Science* **35**(3), 286–303.

Jones, K.L., I.J. Lustig, J.M. Farvolden, and W.B. Powell. 1993. Multicommodity network flows: The impact of formulation on decomposition. *Mathematical Programming,* **62**, 95–117.

Lübbecke, M.E. and J. Desrosiers. 2005. Selected Topics in Column Generation. *Operations Research*, **53**(6), 1007–1023.

Oukil, A., H. Ben Amor, J. Desrosiers, and H. El Gueddari. 2006. Stabilized Column Generation for Highly Degenerate Multiple-Depot Vehicle Scheduling Problems. *Computers & Operations Research*, **34**(3), 817–834.

Pan, P.-Q. 1998. A Basis Deficiency-Allowing Variation of the Simplex Method for Linear Programming. *Computers and Mathematics with Applications*, **36**(3), 33–53.

Raymond, V., F. Soumis, and D. Orban. 2010. An New Version of the Improved Primal Simplex Algorithm for Degenerate Linear Programs. *Computers and Operations Research*, **37**(1), 91–98.

Ryan, D. M. and Osborne, M. 1988. On the Solution to Highly Degenerate Linear Programmes. *Mathematical Programming*, **41**, 385–392.

Stallings, W. 2009. *Computer Organization and Architecture: Designing for Performance*. Prentice Hall.

Terlaky, T. and Sushong, Z. 1993. Pivot Rules for Linear Programming: A Survey on Recent Theoretical Developments. *Annals of Operations Research*, **46-47**(1), 203–233.

Wolfe, P. 1963. A Technique for Resolving Degeneracy in LP. *SIAM Journal*, **11**(2), 205–211.