A linear programming decomposition focusing on the span of the nondegenerate columns

Jérémy Omer, François Soumis

Accepted Manuscript

A linear programming decomposition focusing on the span of the nondegenerate columns

Jérémy Omer, François Soumis

PII: S0377-2217(15)00226-X
DOI: 10.1016/j.ejor.2015.03.019
Reference: EOR 12840

To appear in: *European Journal of Operational Research*

Received date: 28 July 2014
Revised date: 8 March 2015
Accepted date: 12 March 2015

Please cite this article as: Jérémy Omer, François Soumis, A linear programming decomposition focusing on the span of the nondegenerate columns, *European Journal of Operational Research* (2015), doi: 10.1016/j.ejor.2015.03.019

Highlights

- We identify the limits of the existing Improved Primal Simplex (IPS).
- We revise every step of the dynamic reduction implemented in IPS.
- We show how basic solutions can be built to warm start the solution of each problem.
- A simple and fast procedure can test the potential for improvement of the algorithm.
- The algorithm outperforms IPS and CPLEXs primal simplex on a large benchmark.

A linear programming decomposition focusing on the span of the nondegenerate columns

Jérémy Omer[a,b,*], François Soumis[a,b]

[a] *École Polytechnique de Montréal, 2900 bd. Edouard-Montpetit, Montréal (QC), H3T 1J4 Canada*
[b] *Group for Research in Decision Analysis, HEC Montral, 3000 ch. de la Cte-Sainte-Catherine Montréal (QC), H3T 2A7 Canada*

Abstract

The improved primal simplex (IPS) was recently developed by Elhalaloui et al. to take advantage of degeneracy when solving linear programs with the primal simplex. It implements a dynamic constraint reduction based on the compatible columns, i.e., those that belong to the span of a given subset of basic columns including the nondegenerate ones. The identification of the compatible variables may however may be computationally costly and a large number of linear programs are solved to enlarge the subset of basic variables. In this article, we first show how the positive edge criterion of Raymond et al. can be included in IPS for a fast identification of the compatible variables. Our algorithm then proceeds through a series of reduction and augmentation phases until optimality is reached. In a reduction phase, we identify compatible variables and focus on them to make quick progress toward optimality. During an augmentation phase, we compute one greatest normalized improving direction and select a subset of variables that should be considered in the reduced problem. Compared with IPS, the linear program that is solved to find this direction involves the data of the original constraint matrix. This new algorithm is tested over Mittelmann's benchmark for linear programming and on instances arising from industrial applications. The results show that the new algorithm outperforms the primal simplex of CPLEX on most highly degenerate instances in which a sufficient number of nonbasic variables are compatible. In contrast, IPS has difficulties on the eleven largest Mittelmann instances.

Keywords: Linear programming, Degeneracy, Improved primal simplex, Decomposition, Primal algorithms

1. Introduction

We consider a linear program (LP) in standard form:

$$\begin{cases} \min & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0}, \end{cases} \tag{P}$$

where $\boldsymbol{x}, \boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^m$, and $\boldsymbol{A} \in \mathbb{R}^{m\times n}$. We assume that $\boldsymbol{A}$ is of full rank m with $m \leq n$ and that the feasible domain $\mathcal{F}_{\mathrm{P}} = \{\boldsymbol{x} \geq \boldsymbol{0} : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\}$ is nonempty. A basis is a set of m independent columns of $\boldsymbol{A}$, and the associated variables are said to be basic. Starting from the indices $\mathcal{B}$ of the basic variables and $\mathcal{N}$ of the remaining nonbasic variables, the associated basic solution is obtained by setting

$$\boldsymbol{x}_{\mathcal{B}} = \boldsymbol{A}_{.\mathcal{B}}^{-1}\boldsymbol{b} \quad \text{and} \quad \boldsymbol{x}_{\mathcal{N}} = \boldsymbol{0}, \tag{1}$$

where for any set of indices $\mathcal{J}$, $\boldsymbol{A}_{.\mathcal{J}}$ is the set of columns of $\boldsymbol{A}$ indexed by $\mathcal{J}$, and $\boldsymbol{x}_{\mathcal{J}}$ is the corresponding subvector of variables. More generally, the submatrix of $\boldsymbol{A}$ containing the rows indexed by $\mathcal{I}$ and the columns indexed by $\mathcal{J}$ will be denoted $\boldsymbol{A}_{\mathcal{I}\mathcal{J}}$. The basic solution is feasible if and only if $\boldsymbol{x}_{\mathcal{B}} \geq \boldsymbol{0}$. If $\{j \in \mathcal{B} : x_j = 0\}$ is not empty, the solution is said to be degenerate, and all the variables indexed by this set are degenerate. The remaining nonzero basic variables are the nondegenerate variables.

*Tel: +1 (514)340-5121 #6051
Email addresses: jeremy.omer@gerad.ca (Jérémy Omer), francois.soumis@gerad.ca (François Soumis)

1.1. Dealing with degeneracy in the primal simplex

Starting from a basic feasible solution, the primal simplex algorithm (see [4]) monotonically improves the objective value by going through a sequence of neighboring feasible bases until optimality is reached. One theoretical limitation of the algorithm is that an iteration may not lead to any progress in the objective value if the solution is degenerate. Geometrically, a degenerate vertex of the n-dimensional feasible polytope of an LP is the intersection of more than n constraints of this LP. In terms of the simplex algorithm, this means that a single vertex can correspond to several bases. The difficulty is that sometimes many iterations move from one basis to another associated with the same vertex. As a consequence, the theoretical convergence of the simplex cannot be guaranteed without a pivoting rule such as those described in [2, 3].

Although cycling is rarely an issue in practice, the risk of stalling is real. Several techniques have been developed to limit the negative effects of degeneracy [1, 3, 9, 10, 13], but recently there has been a growing interest in methods that take advantage of degeneracy. These studies all rely on the idea that degeneracy corresponds to a local excess of information, since degenerate basic variables are not needed to characterize a vertex of the polytope. Perold [22] exploits this to develop a degeneracy structure in the LU decomposition of the basis, which involves fewer calculations when performing degenerate pivots. Pan [18] took another important step in this direction by generalizing the concept of a basis. He defines a deficient basis to be a set of less than m independent columns of $\boldsymbol{A}$ whose range contains $\boldsymbol{b}$. If the current solution is degenerate, it is sufficient to consider the deficient basis that contains only the positive variables. Degeneracy therefore becomes a potential opportunity to solve smaller linear systems at each iteration. Using deficient bases, Pan develops a simplex-like algorithm [20] and a dual projective algorithm [19] that show promising results in an experimental comparison with MINOS 5 [16].

Elhallaoui et al. [8, 7] also take advantage of degeneracy to speed up the solution of set partitioning problems by aggregating the original constraints into clusters of constraints. The feasibility of the solution is ensured by keeping only the variables that are compatible with the clusters, i.e., the variables that are either present in or absent from every constraint of each cluster. When no improvement can be made by considering the compatible variables, some clusters are broken up or combined to include new improving directions in the aggregated problem. The strength of this dynamic constraint aggregation is the focus on a problem with many fewer constraints than the original one. The improved primal simplex (IPS) [6] extends this approach to general linear programming. A reduced problem is formed by keeping only the nondegenerate and compatible variables. In this context, a variable is compatible when the corresponding column of $\boldsymbol{A}$ is in the range of the p nondegenerate columns. With the incompatible variables removed, $m - p$ constraints are redundant and thus ignored. Once the optimal solution of the reduced problem has been found, a complementary problem is solved to prove the optimality of the original LP or to identify a sequence of pivots ending with an improvement in the objective value. The authors report that IPS significantly outperforms CPLEX[1] on flight assignment (FA), combined vehicle and crew scheduling (VCS), and uncapacitated facility location (UFL) problems, and Raymond et al. [24] describe implementation techniques that improve the performance of the algorithm.

One important limitation of IPS is that compatible variables are identified through costly algebraic operations similar to those performed when computing a simplex tableau. As highlighted by Omer et al. [17], these operations are also useful when solving the complementary problem since they allow us to search for an improving direction in a reduced space, as is done in reduced gradient methods [15]. However, their tests on a diversified benchmark show that these operations cause IPS not to perform well on every highly degenerate LP. Raymond et al. [23] address this issue with a stochastic test requiring as many operations as the computation of a reduced cost to identify all the compatible variables. The authors apply this test to develop a partial pricing algorithm focusing on the compatible variables first. They report good results on the aforementioned VCS and FA instances, but their procedure struggles with two families of instances represented in Mittelmann's benchmark. Based on this test, Towhidi et al. [26] implement the positive edge pricing criterion within COIN-OR LP solver[2] (CLP). Their results show significant improvement with regards to the devex pricing criterion [14] for the most degenerate Mittelmann instances, but their comparison focuses on CLP, which is known to be less efficient than most commercial LP solvers. More importantly, the articles by Towhidi et

[1] CPLEX is freely available for academic and research purposes under the IBM academic initiative: `http://www-03.ibm.com/ibm/university/academic`

[2] `https://projects.coin-or.org/Clp`

al. [26] and Raymond et al. [23] show how a fast compatibility test can be used to cope with degeneracy, but they do not take advantage of degeneracy, since the size of the linear system solved at each simplex pivot is not reduced.

1.2. Contribution statement

Although the dual simplex and barrier algorithms often solve LPs more efficiently than the primal simplex, the latter has a strong advantage when a good feasible solution is available. As a consequence, the primal simplex is still used for reoptimization after modifications in the objective function, or after adding columns in the master problem in a column-generation procedure. Our work thus focuses on improving the primal simplex by taking advantage of degeneracy.

Our main contribution is a new dynamic reduction algorithm that overcomes the difficulties that IPS encounters on large instances. This algorithm not only yields substantial improvement on many degenerate instances but also provides a fast procedure to test the potential for improvement in advance. To achieve this, we modify IPS to include the fast compatibility test described in [23]. One negative effect is that the complementary problem cannot be reduced without doing the algebraic operations that we are trying to avoid. The algorithm thus focuses on a complementary problem involving the original constraints of P, and it involves a new mode of alternation between the reduced and the complementary problems that is more efficient on large LPs. We then show how good basic solutions can be built to warm-start both the reduced and the complementary problems. The practical impact of these modifications is studied on a large benchmark including the VCS, FA, and UFL instances used in [17] and fourty-five Mittelmann instances. The purpose is to evaluate our new algorithm by comparing it with IPS and the primal simplex of CPLEX, and to show that it is possible to identify quickly the instances that offer a strong potential for faster solution. The results show the potential of the algorithm for an implementation as an adaptive strategy in a state-of-the-art primal simplex code.

In Section 2 we describe IPS as a necessary background for the rest of the article. The new algorithm based on a fast compatibility test is developed in Section 3. The results of the experimental tests are presented and analyzed in Section 4, and in Section 5 we discuss directions for future research.

2. The improved primal simplex

In this section, we summarize the theoretical foundations and the practical implementation of IPS as a background for the new algorithm developped in Section 3. Although efficient implementations of linear programming algorithms should focus on LPs with bounded variables, we consider an LP in standard form to clarify and shorten the presentation. Omer et al. [17] show the generalization to an LP with bounded variables, and the implementations tested in Section 4 use this generalization.

Let $\boldsymbol{x} \in \mathcal{F}_{\mathrm{P}}$ be a basic feasible solution of P. The variables' indices can be partitioned into two sets $\mathcal{P} = \{j : x_j > 0\}$ and $\mathcal{L} = \{j : x_j = 0\}$. Since $\boldsymbol{x}$ is a basic solution, the variables indexed by $\mathcal{P}$ are basic, and the columns of $\boldsymbol{A}_{.\mathcal{P}}$ are linearly independent. The cardinality of $\mathcal{P}$, $p = |\mathcal{P}|$, thus satisfies $p \leq m$. Moreover, $\boldsymbol{x}_{\mathcal{L}} = \boldsymbol{0}$, so the range of $\boldsymbol{A}_{.\mathcal{P}}$ contains $\boldsymbol{b}$.

Remark 1. *If $p < m$, $\boldsymbol{A}_{.\mathcal{P}}$ satisfies the definition of a deficient basis given by Pan [18].*

2.1. A primal decomposition for degenerate problems

Assuming that $\boldsymbol{x} \in \mathcal{F}_{\mathrm{P}}$, a primal algorithm iteratively improves the objective value by following a sequence of *feasible improving directions* as introduced below.

Definition 1 (feasible direction)**.** *$\boldsymbol{d} \in \mathbb{R}^n$ is a feasible direction at $\boldsymbol{x}$ if there exists $\rho > 0$ such that $\boldsymbol{x} + \rho \cdot \boldsymbol{d} \in \mathcal{F}_{\mathrm{P}}$.*

Definition 2 (improving direction)**.** *$\boldsymbol{d} \in \mathbb{R}^n$ is an improving direction if $\boldsymbol{c}^T\boldsymbol{d} < 0$, i.e., if taking a positive step along $\boldsymbol{d}$ yields an improvement in the objective value.*

An improving vector is characterized by any pair $(\boldsymbol{d}, \rho)$ such that $\boldsymbol{d}$ is a feasible improving direction and $\rho > 0$. $\boldsymbol{d}$ can thus be normalized at will as long as ρ is chosen to satisfy $\boldsymbol{x} + \rho \cdot \boldsymbol{d} \in \mathcal{F}_{\mathrm{P}}$. One way to make a step toward optimality

is to follow a normalized feasible improving direction that maximizes the rate of improvement in the objective value. The *greatest normalized improvement* (GNI) program finds one of these directions:

$$\begin{cases} \min\limits_{\boldsymbol{d}} & \boldsymbol{c}^T\boldsymbol{d} \\ \text{s.t.} & \boldsymbol{A}\boldsymbol{d} = \boldsymbol{0} \\ & \boldsymbol{w}_{\mathcal{L}}^T\boldsymbol{d}_{\mathcal{L}} \leq 1 \\ & \boldsymbol{d}_{\mathcal{L}} \geq 0, \end{cases} \tag{GNI}$$

where $\boldsymbol{w}_{\mathcal{L}} > 0$ is a normalization vector.

Proposition 1. GNI *has an optimal solution* $\boldsymbol{d}^\star$. *Moreover, denoting* $\mathcal{I}^- = \{i \in \mathcal{P} : d_i^\star < 0\}$,

- $\boldsymbol{x}$ *is an optimal solution of* P *if and only if* $\boldsymbol{c}^T\boldsymbol{d}^\star = 0$*;*
- $\boldsymbol{d}^\star$ *is an optimal ray of* P *if and only if* $\boldsymbol{c}^T\boldsymbol{d}^\star < 0$ *and* $\mathcal{I}^- = \emptyset$*;*
- *if* $\boldsymbol{c}^T\boldsymbol{d}^\star < 0$ *and* $\mathcal{I}^- \neq \emptyset$*, the maximal feasible step along* $\boldsymbol{d}^\star$ *is performed by setting* $\boldsymbol{x} \leftarrow \boldsymbol{x} + \rho^{\max} \cdot \boldsymbol{d}^\star$*, where* $\rho^{\max} = \min\limits_{i \in \mathcal{I}^-} \left\{ \dfrac{x_i}{d_i} \right\}$*. This step leads to an improvement* $\rho^{\max}\boldsymbol{c}^T\boldsymbol{d}^\star$ *in the objective value.*

Proof. $\boldsymbol{d} = \boldsymbol{0}$ is feasible, so GNI is feasible. Assuming that GNI is unbounded, it admits a feasible ray $\boldsymbol{r}$ with $\boldsymbol{c}^T\boldsymbol{r} < 0$. $\forall \lambda \geq 0, \lambda \cdot \boldsymbol{r}$ is feasible, so $\lambda \boldsymbol{w}_{\mathcal{L}}^T\boldsymbol{r}_{\mathcal{L}} \leq 1$, with $\boldsymbol{r}_{\mathcal{L}} \geq 0$ and $\boldsymbol{w}_{\mathcal{L}} > 0$, hence $\boldsymbol{r}_{\mathcal{L}} = \boldsymbol{0}$. It follows that $\boldsymbol{A}_{\cdot\mathcal{P}}\boldsymbol{r}_{\mathcal{P}} = \boldsymbol{0}$, with $\boldsymbol{A}_{\cdot\mathcal{P}}$ of full rank, so $\boldsymbol{r}_{\mathcal{P}} = \boldsymbol{0}$. This contradicts $\boldsymbol{c}^T\boldsymbol{r} > 0$. GNI is thus bounded and has an optimal solution $\boldsymbol{d}^\star$.

Next, assume that $\boldsymbol{c}^T\boldsymbol{d}^\star < 0$. $\boldsymbol{d}^\star$ is an optimal ray of P if and only if $\boldsymbol{d}^\star \geq \boldsymbol{0}$, or equivalently $\Leftrightarrow \mathcal{I}^- = \emptyset$. If $\mathcal{I}^- \neq \emptyset$, it is easy to verify that $\rho^{\max} > 0$ is the largest step such that $\boldsymbol{x} + \rho^{\max}\boldsymbol{d}^\star \geq \boldsymbol{0}$ is feasible, and that it leads to an improvement $\rho^{\max}\boldsymbol{c}^T\boldsymbol{d}^\star < 0$ in the objective value. $\boldsymbol{x}$ is not an optimal solution of P, so $\boldsymbol{x}$ optimal $\Rightarrow \boldsymbol{d}^\star \geq 0$.

Finally, if $\boldsymbol{c}^T\boldsymbol{d}^\star \geq 0$, no improving feasible direction exists at $\boldsymbol{x}$, so $\boldsymbol{x}$ is an optimal solution of P if and only if $\boldsymbol{c}^T\boldsymbol{d}^\star = 0$. □

To take advantage of degeneracy, Omer et al. [17] decompose GNI by referring to the following concept of compatibility.

Definition 3 (compatible variable)**.** *A variable* x_j *is compatible with a deficient basis* $\mathcal{P}$ *if and only if the corresponding column in* $\boldsymbol{A}$*,* $\boldsymbol{a}_j$*, is in* Span$\left(\boldsymbol{A}_{\cdot\mathcal{P}}\right)$.

The zero variables can then be partitioned into compatible and incompatible variables whose sets of indices are respectively denoted $\mathcal{C}$ and $\mathcal{I}$. Based on this partition, GNI is decomposed into the reduced-GNI (R-GNI) and the complementary-GNI (C-GNI) that respectively focus on the compatible and incompatible variables:

$$\begin{array}{lrcl} \min\limits_{(\boldsymbol{d}_{\mathcal{P}}, \boldsymbol{d}_C)} & \boldsymbol{c}_{\mathcal{P}}^T\boldsymbol{d}_{\mathcal{P}} + \boldsymbol{c}_C^T\boldsymbol{d}_C & & \\ \text{s.t.} & \boldsymbol{A}_{\cdot\mathcal{P}}\boldsymbol{d}_{\mathcal{P}} + \boldsymbol{A}_{\cdot C}\boldsymbol{d}_C & = & 0 \\ & \boldsymbol{w}_C^T\boldsymbol{d}_C & \leq & 1 \\ & \boldsymbol{d}_C \geq 0 & & \end{array} \tag{R-GNI}$$

$$\begin{array}{lrcl} \min\limits_{(\boldsymbol{d}_{\mathcal{P}}, \boldsymbol{d}_{\mathcal{I}})} & \boldsymbol{c}_{\mathcal{P}}^T\boldsymbol{d}_{\mathcal{P}} + \boldsymbol{c}_{\mathcal{I}}^T\boldsymbol{d}_{\mathcal{I}} & & \\ \text{s.t.} & \boldsymbol{A}_{\cdot\mathcal{P}}\boldsymbol{d}_{\mathcal{P}} + \boldsymbol{A}_{\cdot\mathcal{I}}\boldsymbol{d}_{\mathcal{I}} & = & 0 \\ & \boldsymbol{w}_{\mathcal{I}}^T\boldsymbol{d}_{\mathcal{I}} & \leq & 1 \\ & \boldsymbol{d}_{\mathcal{I}} \geq 0. & & \end{array} \tag{C-GNI}$$

The validity of this decomposition is justified by the following theorem of Omer et al. [17], which extends to general LPs a result proved by Rosat et al. [25] for set partitioning problems.

Theorem 1. *Let* $z^\star_{\text{GNI}}$, $z^\star_{\text{R-GNI}}$, *and* $z^\star_{\text{C-GNI}}$ *be the optimal objective values of* GNI, R-GNI, *and* C-GNI*; then we have*

$$z^\star_{\text{GNI}} = \min\left\{z^\star_{\text{R-GNI}}, z^\star_{\text{C-GNI}}\right\}.$$

Theorem 1 ensures that an optimal solution of GNI can be found by solving R-GNI and C-GNI separately.

One key feature of the decomposition is that R-GNI can be further reduced. If the current solution is degenerate, i.e., if $p < m$, the rank of the constraint matrix of R-GNI (p) is lower than the number of constraints (m), so $m-p$ constraints of R-GNI are redundant. Assume that the rows and columns are permuted so that the first p columns correspond to the nonzero variables and the first p rows correspond to independent rows of $\boldsymbol{A}_{\cdot\mathcal{P}}$; then R-GNI is equivalent to

$$\begin{array}{lllllll} \min\limits_{(\boldsymbol{d}_\mathcal{P}, \boldsymbol{d}_C)} & \boldsymbol{c}_\mathcal{P}^T \boldsymbol{d}_\mathcal{P} & + & \boldsymbol{c}_C^T \boldsymbol{d}_C & & \\ \text{s.t.} & \boldsymbol{A}_{\mathcal{P}\mathcal{P}} \boldsymbol{d}_\mathcal{P} & + & \boldsymbol{A}_{\mathcal{P}C} \boldsymbol{d}_C & = & 0 \\ & & & \boldsymbol{w}_C^T \boldsymbol{d}_C & \leq & 1 \\ & & & \boldsymbol{d}_C \geq 0 & & \end{array}$$

The practical benefit of the decomposition is that an efficient solution process can focus on the easy subproblem R-GNI until $z^\star_{\text{R-GNI}} = 0$, and then solve C-GNI to find an improving direction involving several incompatible variables or to prove the optimality of the current solution.

2.2. A practical implementation: IPS

One important step of the decomposition is the identification of the compatible variables. In IPS, the deficient basis $\boldsymbol{A}_{\cdot\mathcal{P}}$ is completed with $m - p$ independent vectors to form a basis $\boldsymbol{B}$ of $\mathbb{R}^m$. Let $\overline{\mathcal{P}} = \{1, \ldots, m\} \setminus \mathcal{P}$. The compatible variables are then identified using the following result.

Proposition 2. $\boldsymbol{v} \in \mathbb{R}^m$ *is compatible with* $\mathcal{P}$ *if and only if* $\left(\boldsymbol{B}^{-1}\boldsymbol{v}\right)_{\overline{\mathcal{P}}} = \boldsymbol{0}$.

Proof. $\boldsymbol{v} \in \mathbb{R}^m$ is written uniquely in the basis $\boldsymbol{B}$ as $\left(\left(\boldsymbol{B}^{-1}\boldsymbol{v}\right)_\mathcal{P}, \left(\boldsymbol{B}^{-1}\boldsymbol{v}\right)_{\overline{\mathcal{P}}}\right)$. $\left(\boldsymbol{B}^{-1}\boldsymbol{v}\right)_\mathcal{P} \in \text{Span}(\boldsymbol{A}_{\cdot\mathcal{P}})$ gives the coordinates of $\boldsymbol{v}$ corresponding to $\boldsymbol{A}_{\cdot\mathcal{P}}$, so $\boldsymbol{v} \in \text{Span}(\boldsymbol{A}_{\cdot\mathcal{P}})$ if and only if $\left(\boldsymbol{B}^{-1}\boldsymbol{v}\right)_{\overline{\mathcal{P}}} = \boldsymbol{0}$. □

A computationally efficient completion of $\boldsymbol{A}_{\cdot\mathcal{P}}$ uses vectors of the canonical basis of $\mathbb{R}^m$. Assuming a convenient reordering of the rows, $\boldsymbol{B}$ and its inverse are given by

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{A}_{\mathcal{P}\mathcal{P}} & \boldsymbol{0} \\ \boldsymbol{A}_{\overline{\mathcal{P}}\mathcal{P}} & \boldsymbol{I}_{m-p} \end{bmatrix} \Leftrightarrow \boldsymbol{B}^{-1} = \begin{bmatrix} \boldsymbol{A}_{\mathcal{P}\mathcal{P}}^{-1} & \boldsymbol{0} \\ -\boldsymbol{A}_{\overline{\mathcal{P}}\mathcal{P}}\boldsymbol{A}_{\mathcal{P}\mathcal{P}}^{-1} & \boldsymbol{I}_{m-p} \end{bmatrix}.$$

The compatible variables of $\mathcal{L}$ are thus identified by searching for the zero columns of

$$\bar{\boldsymbol{A}} = -\boldsymbol{A}_{\overline{\mathcal{P}}\mathcal{P}}\boldsymbol{A}_{\mathcal{P}\mathcal{P}}^{-1}\boldsymbol{A}_{\mathcal{P}\mathcal{L}} + \boldsymbol{A}_{\overline{\mathcal{P}}\mathcal{L}}.$$

$\bar{\boldsymbol{A}}$ is a restriction of the simplex tableau to the $m - p$ rows of $\overline{P}$, and it is well known that the tableau version of the simplex had to be abandoned because of these algebraic operations. A practical implementation of IPS thus requires us to limit the number of times that the compatible variables are identified, and to take advantage of the knowledge of $\bar{\boldsymbol{A}}$.

Omer et al. [17] show that solving R-GNI and following the resulting improving direction is equivalent to performing a simplex pivot in the original LP restricted to the nondegenerate and compatible variables, P_C. The redundant constraints of P_C are identified by performing a LU decomposition of $\boldsymbol{A}_{\cdot\mathcal{P}}$ using UMFPACK [5]. Denoting $\boldsymbol{U}$ the upper triangle matrix provided by the decomposition, a set of dependent rows of $\boldsymbol{A}_{\cdot\mathcal{P}}$ is deduced from the rows of $\boldsymbol{U}$ with at least one nonzero element. After removing the redundant constraints, this reduced LP is

$$\left\{\begin{array}{ll} \min & \boldsymbol{c}_\mathcal{P}^T \boldsymbol{x}_\mathcal{P} + \boldsymbol{c}_C^T \boldsymbol{x}_C \\ \text{s.t.} & \boldsymbol{A}_{\mathcal{P}\mathcal{P}} \boldsymbol{x}_\mathcal{P} + \boldsymbol{A}_{\mathcal{P}C} \boldsymbol{x}_C = \boldsymbol{b}_\mathcal{P} \\ & \boldsymbol{x}_\mathcal{P} \geq \boldsymbol{0}, \boldsymbol{x}_C \geq \boldsymbol{0} \end{array}\right. \qquad (\text{P}_C)$$

The variable pivoted into the basis minimizes the normalized pricing criterion $\bar{c}_i/w_i$, where $\bar{\boldsymbol{c}}^T = \boldsymbol{c}^T - \boldsymbol{c}_{\mathcal{P}}^T \boldsymbol{A}_{\mathcal{P}\mathcal{P}}^{-1} \boldsymbol{A}_{\mathcal{P}C}$. As a consequence, if the partition $(\mathcal{P}, C, \mathcal{I})$ is not updated after the solution of R-Gni, this problem may be solved several times by performing as many simplex pivots in P_C. The drawback is that some basic variables may take a zero value, in which case degenerate pivots may occur. However, this risk is compensated for by the time saved on the update of $\bar{\boldsymbol{A}}$. Unless the optimal solution is found, the solution of P_C is interrupted after an arbitrary number of pivots, and $(\mathcal{P}, C, \mathcal{I})$ is updated if two conditions that formulate the expected benefit of a new partition are satisfied. In [17], IPS interrupts the solution of P_C every m iterations, and in order to update $(\mathcal{P}, C, \mathcal{I})$ it requires that at least 10% of the basic variables of P_C are degenerate and 30% of the nondegenerate variables do not belong to $\mathcal{P}$.

Once an optimal solution of P_C has been found, C-Gni is solved to find a new improving direction or to prove the optimality of the current solution. In C-Gni, the linear constraints $\boldsymbol{A}_{\cdot\mathcal{P}}\boldsymbol{d}_{\mathcal{P}} + \boldsymbol{A}_{\cdot\mathcal{I}}\boldsymbol{d}_{\mathcal{I}} = \boldsymbol{0}$ can be interpreted as follows: the weighted combination of the columns $\boldsymbol{A}_{\cdot\mathcal{I}}\boldsymbol{d}_{\mathcal{I}}$ that potentially enter the basis must be compatible. According to Proposition 2, another way of putting this condition is $\bar{\boldsymbol{A}}_{\cdot\mathcal{I}}\boldsymbol{d}_{\mathcal{I}} = \boldsymbol{0}$. For a given $\boldsymbol{d}_{\mathcal{I}}$ satisfying this condition, the unique vector $\boldsymbol{d}_{\mathcal{P}}$ satisfying $\boldsymbol{A}_{\cdot\mathcal{P}}\boldsymbol{d}_{\mathcal{P}} + \boldsymbol{A}_{\cdot\mathcal{I}}\boldsymbol{d}_{\mathcal{I}} = \boldsymbol{0}$ is then given by $\boldsymbol{d}_{\mathcal{P}} = -\boldsymbol{A}_{\mathcal{P}\mathcal{P}}^{-1}\boldsymbol{A}_{\mathcal{P}\mathcal{I}}\boldsymbol{d}_{\mathcal{I}}$. Since $\bar{\boldsymbol{A}}_{\cdot\mathcal{I}}$ was computed when the compatible variables were identified, C-Gni can be modified without additional operations to search for an optimal solution by solving a smaller LP that involves only the variables of $\boldsymbol{d}_{\mathcal{I}}$:

$$
\begin{array}{rlll}
z^{\star}_{\text{C-Gni}} = & \min\limits_{\boldsymbol{d}_{\mathcal{I}}} & \bar{\boldsymbol{c}}_{\mathcal{I}}^T \boldsymbol{d}_{\mathcal{I}} & \\
& \text{s.t.} & \bar{\boldsymbol{A}}_{\cdot\mathcal{I}}\boldsymbol{d}_{\mathcal{I}} = 0 & \\
& & \boldsymbol{w}_{\mathcal{I}}^T \boldsymbol{d}_{\mathcal{I}} \leq 1 & \\
& & \boldsymbol{d}_{\mathcal{I}} \geq 0 &
\end{array}
\qquad \text{(Cr-Gni)}
$$

Algorithm 1: IPS

Data: P, a linear program, $\boldsymbol{x}^0$, a basic feasible solution of P.
Result: $\boldsymbol{x}$, an optimal solution of P.

begin
  $\boldsymbol{x} \leftarrow \boldsymbol{x}^0$; $\mathcal{P} \leftarrow$ the basis associated to $\boldsymbol{x}^0$; $\mathcal{B}^+ \leftarrow \{j : x_j^0 > 0\}$; Red $\leftarrow$ P; *optimal*$\leftarrow$ *false*;
  while *optimal* = *false* **do**
    while Red *is not solved to optimality* **do**
      if $|\mathcal{B}^+| \leq 0.9\,|\mathcal{P}|$ *and* $|\mathcal{P} \cap \mathcal{B}^+| \leq 0.7\,|\mathcal{P}|$ **then**
        Compute $\bar{\boldsymbol{A}}$ to build the partition $(\mathcal{P}, C, \mathcal{I})$ corresponding to $\boldsymbol{x}$;
        Red $\leftarrow \mathrm{P}_C$;
      Solve Red for m iterations or until optimality;
      if Red *is found to be unbounded* **then**
        Complete an optimal ray of Red to get an optimal ray, $\boldsymbol{x}$, of P; *optimal*$\leftarrow$ *true*;
      else
        Update $\boldsymbol{x}$; $\mathcal{B}^+ \leftarrow \{j : x_j > 0\}$;
    Solve Cr-Gni: $\boldsymbol{d}^{\star}_{\mathcal{I}} \leftarrow$ the optimal solution; $z^{\star}_{\text{Cr-Gni}} \leftarrow$ the optimal objective value;
    if $z^{\star}_{\text{Cr-Gni}} \geq 0$ **then**
      optimal $\leftarrow$ *true*;
    else
      $\mathcal{I}^+ \leftarrow \{i \in \mathcal{I} : d_i^{\star} > 0\}$;
      while $z^{\star}_{\text{Cr-Gni}} < 0$ *and* $|\mathcal{I}^+| \leq 0.1\,|\mathcal{I}|$ **do**
        Remove $\boldsymbol{d}_{\mathcal{I}^+}$ from Cr-Gni;
        Solve Cr-Gni and update $\boldsymbol{d}^{\star}_{\mathcal{I}}$ and $z^{\star}_{\text{Cr-Gni}}$;
        $\mathcal{I}^+ \leftarrow \mathcal{I}^+ \cup \{i \in \mathcal{I} : d_i^{\star} > 0\}$; ;
      Add the columns indexed by $\mathcal{I}^+$ and the newly compatible ones to Red;

Let $\boldsymbol{d}^{\star}_{\mathcal{I}}$ be an optimal solution of Cr-Gni. If $(\mathcal{P}, C, \mathcal{I})$ is not updated before Cr-Gni is solved, the compatibility of the combination $\boldsymbol{A}_{\cdot\mathcal{I}}\boldsymbol{d}_{\mathcal{I}}$ does not necessarily imply that the optimal solution of Cr-Gni corresponds to a feasible

direction for P. This is not a real issue in IPS, because this solution is not used to produce a strict improvement in the objective value but instead to select columns that should be added to the reduced problem. Since $\boldsymbol{d}^{\star}_{\mathcal{I}}$ indicates an interesting direction in most cases, we select every variable indexed by $\mathcal{I}^{+} = \{j \in \mathcal{I} : d^{\star}_{j} > 0\}$. Cr-Gni is then updated by removing the variables indexed by $\mathcal{I}^{+}$ and solved several times until at least 10% of the variables of $\mathcal{I}$ are selected.

Algorithm 1 summarizes the overall procedure. In this procedure, IPS solves P using a dynamic reduction of the problem. The algorithm performs major iterations (step 4–22) composed of a reduction phase (step 4–12) and an augmentation phase (step 13–22), until optimality is reached. At each major iteration, IPS performs simplex pivots on the reduced problem Red, and, depending on the state of the solution, the number of rows and columns in Red may increase and/or decrease. The motivation for developing the dynamic reduction is that the solution may be accelerated if the majority of the pivots can be performed on a problem smaller than the original one.

In this implementation, Red is always set to P_{C} after an update of $(\mathcal{P}, C, \mathcal{I})$, and it is gradually augmented with new columns after each solution of the complementary problem. Rows are added together with the columns to ensure that the constraints of P that do not appear in Red are the redundant ones (this does not appear explicitly in the algorithm).

To ensure that the interruptions and subsequent warm starts are done efficiently, every LP is solved with a simplex algorithm. To be specific, Red is solved with the primal simplex and Cr-Gni is solved with the dual simplex. Raymond et al. [24] justify the choice of the dual simplex for Cr-Gni by a large number of computational tests. A more qualitative reason is that the initial solution of Cr-Gni is usually degenerate. Using the dual simplex offers an opportunity to escape from a highly degenerate vertex without going through a long sequence of degenerate pivots. From this perspective, IPS is a primal–dual algorithm that switches to the dual simplex when no improvement can be achieved in the subspace spanned by the variables of $\mathcal{P}$.

3. IPS with original constraints in the complementary problem: IPSO

In this section, we develop a new dynamic reduction process to include a fast compatibility test that does not require the computation of $\bar{\boldsymbol{A}}$. This new algorithm, called IPSO, maintains the global structure of IPS: reduction and augmentation phases are iteratively performed until optimality is reached. However, the reduction and augmentation phases are both revised to operate with no access to the information contained in $\bar{\boldsymbol{A}}$. To characterize the differences between IPS and IPSO, we may bring out an analogy with the differences between Dantzig's primal simplex and the revised simplex method. In its initial description, the primal simplex computes the simplex tableau $\bar{\boldsymbol{A}}$ and updates it with each pivot. To avoid spending too much time in the computation of $\bar{\boldsymbol{A}}$, the revised simplex records only the indices of the basic variables and it solves two linear systems to identify the entering and exiting variables at each iteration. Similarly to the revised simplex, IPSO records only the indices of the compatible variables and updates them by solving linear systems. One issue with this new process is that IPS also uses $\bar{\boldsymbol{A}}$ to reduce the complementary problem. As a consequence, one important difference between IPSO and IPS is that the augmentation phase is completely redesigned to limit the time spent in the solution of the complementary problem.

Algorithm 2 summarizes the new dynamic reduction process. The key features of IPSO include the adaptation of the positive edge criterion for fast identification of the compatible variables (steps 6 and 13) and a heuristic augmentation procedure that requires only one solution of the complementary problem at each phase (steps 13–27). Warm-starting the reduced and complementary problems is another essential contribution for an efficient transition between the phases (steps 16, 19 and 27). These important steps of the algorithm are detailed below.

3.1. Building the partition

In steps 6 and 13 of Algorithm 2, we adapt the positive edge criterion of Raymond et al. [23] for a fast identification of the compatible variables. More generally, this criterion can be viewed as a stochastic test of membership in the range of a set of independent vectors. As for IPS, the authors of [23] complete the deficient basis $\boldsymbol{A}_{\cdot\mathcal{P}}$ with vectors of the canonical basis of $\mathbb{R}^m$ to form a basis $\boldsymbol{B}$ of $\mathbb{R}^m$. Let $j \in \{1, \ldots, n\}$, and $\bar{\boldsymbol{a}}_j = \left(\boldsymbol{B}^{-1}\boldsymbol{a}_j\right)_{\overline{\mathcal{P}}}$. The essence of the positive edge criterion is then given by the following proposition.

Proposition 3. *Let $\boldsymbol{v}$ be a vector of $m - p$ continuous random variables. Then, either $\boldsymbol{a}_j$ is compatible with $\mathcal{P}$ and $\boldsymbol{v}^T \cdot \bar{\boldsymbol{a}}_j = 0$, or $\boldsymbol{a}_j$ is incompatible and $\mathbb{P}\left(\boldsymbol{v}^T\bar{\boldsymbol{a}}_j = 0\right) = 0$.*

Algorithm 2: IPSO

Data: P, a linear program, $\boldsymbol{x}^0$, a basic feasible solution of P.
Result: $\boldsymbol{x}$, an optimal solution of P.

begin
  $\boldsymbol{x} \leftarrow \boldsymbol{x}^0$; $\mathcal{P} \leftarrow$ the basis associated with $\boldsymbol{x}^0$; $\mathcal{B}^+ \leftarrow \{j : x_j^0 > 0\}$; RED $\leftarrow$ P; *optimal*$\leftarrow$ *false*;
  while *optimal* = *false* **do**
    `/* Reduction and solution of RED */`
    while RED *is not solved to optimality* **do**
      if $|\mathcal{B}^+| \leq 0.9\,|\mathcal{P}|$ *and* $|\mathcal{P} \cap \mathcal{B}^+| \leq 0.7\,|\mathcal{P}|$ **then**
        Use the positive edge criterion to build the partition $(\mathcal{P}_0, \mathcal{C}_0, \mathcal{I}_0)$ corresponding to $\boldsymbol{x}$;
        RED $\leftarrow \mathrm{P}_{\mathcal{C}_0}$; `// reduction of RED`
      Solve RED for m iterations or until optimality;
      if RED *is found to be unbounded* **then**
        Complete an optimal ray of RED to get an optimal ray, $\boldsymbol{x}$, of P; *optimal*$\leftarrow$ *true*;
      else
        Update $\boldsymbol{x} \leftarrow$; $\mathcal{B}^+ \leftarrow \{j : x_j > 0\}$;
    `/* Augmentation of RED */`
    Use the positive edge criterion to build the partition $(\mathcal{P}, \mathcal{C}, \mathcal{I})$ corresponding to $\boldsymbol{x}$;
    if RED *has m rows* **then** `// RED was augmented since last reduction`
      Build GNI according to $(\mathcal{P}, \mathcal{C}, \mathcal{I})$;
      Compute a good dual feasible basis of GNI and solve it from this initial basis;
    else `// RED was not augmented since last reduction`
      Build C-GNI according to $(\mathcal{P}, \mathcal{C}, \mathcal{I})$;
      Compute a good dual feasible basis of C-GNI and solve it from this initial basis;
    $(\boldsymbol{d}^\star, \mathcal{B}^\star) \leftarrow$ the primal solution of the problem just solved and the associated basis;
    $(\boldsymbol{\pi}^\star, z^\star_{\mathrm{GNI}}) \leftarrow$ the dual solution and the objective value of the problem just solved;
    if $z^\star_{\mathrm{GNI}} = 0$ *or* $\boldsymbol{d}^\star \geq \boldsymbol{0}$ **then**
      optimal $\leftarrow$ *true*;
    else
      $\boldsymbol{x} \leftarrow \boldsymbol{x} + \rho^{\max} \boldsymbol{d}^\star$;
      Add every column j such that $c_j - \boldsymbol{a}_j^T \boldsymbol{\pi}^\star < 0$ to RED;
      Add every removed row to RED; `// RED now has` m `rows`
    Build a feasible basis for P using $\mathcal{B}^\star$ and $\boldsymbol{d}^\star$;

Proof. If $\boldsymbol{a}_j$ is compatible with $\mathcal{P}$, Proposition 2 states that $\bar{\boldsymbol{a}}_j = \boldsymbol{0}$, hence $\boldsymbol{v}^T\bar{\boldsymbol{a}}_j = 0$. Otherwise, $\boldsymbol{v}^T\bar{\boldsymbol{a}}_j$ is a nonzero continuous random variable, which implies that the probability it takes a particular value is zero. □

Given $\boldsymbol{v} \in \mathbb{R}^{m-p}$, let $\boldsymbol{w} \in \mathbb{R}^m$ be the solution of $\boldsymbol{w}^T\boldsymbol{B} = \left(\boldsymbol{0}, \boldsymbol{v}^T\right)$, i.e., $\boldsymbol{w}^T = \boldsymbol{v}^T\left(\boldsymbol{B}^{-1}\right)_{\overline{\mathcal{P}}}$. $\boldsymbol{v}^T\bar{\boldsymbol{a}}_j = \boldsymbol{w}^T\boldsymbol{a}_j$, so the compatible variables may be identified by solving one linear system of m equations $\boldsymbol{w}^T\boldsymbol{B} = \left(\boldsymbol{0}, \boldsymbol{v}^T\right)$ and computing one matrix/vector product $\boldsymbol{w}^T\boldsymbol{A}_{\cdot\mathcal{L}}$, which requires the same number of operations as the computation of a reduced cost vector. Compared with the operations performed by IPS, this test divides the complexity of the update of $(\mathcal{P}, C, \mathcal{I})$ by $n - p$.

Remark 2. *In practice, the bit representation of floating points implies that the probability of mistakenly identifying a variable as compatible is greater than zero. This case is problematic if a false-compatible variable is positive in the solution of* Red*, because it may result in this solution being outside $\mathcal{F}_P$. Since the primal feasibility of $\boldsymbol{x}$ is an essential assumption for the convergence of Algorithm 2, it is necessary to check that the solution of* Red *is in $\mathcal{F}_P$. If $\boldsymbol{x} \notin \mathcal{F}_P$, feasibility is recovered by performing a few dual simplex pivots on P_C with every constraint of* P.

3.2. Augmenting the reduced problem

In steps 4 to 12 of Algorithm 2, Red is solved to optimality, and reduced if needed. An augmentation phase is then executed from step 13 to step 27. Two slightly different augmentation phases may be executed depending on whether or not Red has been reduced at step 7 since the last augmentation phase.

3.2.1. Case 1: no augmentation phase has been performed since the last reduction

Since constraints were removed during the reduction, Red includes $|\mathcal{P}_0| < m$ rows. Referring to the test at step 14, the complementary problem C-Gni is then used to select the variables of $\mathcal{I}$ that should be included in Red. Since the positive edge criterion is used to identify the compatible variables, $\bar{\boldsymbol{A}}$ is no longer available to form the reduced complementary problem Cr-Gni. While this implies that the complementary problem may take longer to solve, it also means that $(\mathcal{P}, C, \mathcal{I})$ may be updated before we solve C-Gni. With this update, the following proposition holds.

Proposition 4. *The solution of* C-Gni *at step 19 of Algorithm 2 provides a feasible improving direction for* P *if $z^\star_{\text{C-Gni}} < 0$, or proves that the current solution $\boldsymbol{x}$ is optimal if $z^\star_{\text{C-Gni}} = 0$.*

Proof. Let $(\mathcal{P}_0, C_0, \mathcal{I}_0)$ be the partition computed during the last execution of step 6, $(\mathcal{P}, C, \mathcal{I})$ the partition computed at step 13, and R-Gni and C-Gni the complementary and reduced problems corresponding to $(\mathcal{P}, C, \mathcal{I})$. $\mathcal{P} \subset \mathcal{P}_0 \cup C_0$, so $\boldsymbol{A}_{\cdot\mathcal{P}} \subset \text{Span}\left(\boldsymbol{A}_{\cdot P_0}\right)$ and $C \subset \mathcal{P}_0 \cup C_0$. Therefore, (Red is solved to optimality) $\Rightarrow z^\star_{\text{R-Gni}} = 0$. From Theorem 1, it follows that $z^\star_{\text{Gni}} = z^\star_{\text{C-Gni}}$. Proposition 1 then gives the result. □

In IPSO, to avoid spending too much time in the augmentation phase, we solve the complementary problem just once before focusing again on the reduced problem. Assuming that $z^\star_{\text{C-Gni}} < 0$, the primal solution of C-Gni then provides an improving feasible direction at $\boldsymbol{x}$. Let $(\boldsymbol{d}^\star_{\mathcal{P}}, \boldsymbol{d}^\star_{\mathcal{I}})$ be an optimal solution of C-Gni; we may complete it with zeros to obtain $\boldsymbol{d}^\star$. The solution is then updated via $\boldsymbol{x} \leftarrow \boldsymbol{x} + \rho^{\max} \cdot \boldsymbol{d}^\star$ for an improvement $\rho^{\max} z^\star_{\text{C-Gni}}$ in the objective value, with $\rho^{\max}$ computed as in Proposition 1.

The augmentation itself then relies on the dual solution of C-Gni. Let $(\boldsymbol{\pi}, \lambda)$ be the dual variables of C-Gni. The dual of the complementary problem is

$$\begin{array}{lll} \max\limits_{(\boldsymbol{\pi},\lambda)} & \lambda & \\ \text{s.t.} & \boldsymbol{a}_j^T\boldsymbol{\pi} + w_j\lambda \le c_j, & \forall j \in \mathcal{I} \\ & \boldsymbol{a}_j^T\boldsymbol{\pi} = c_j, & \forall j \in \mathcal{P} \\ & \lambda \le 0 & \end{array} \tag{Cd-Gni}$$

Proposition 5. *Let $(\boldsymbol{\pi}^\star, \lambda^\star)$ be the optimal solution of* Cd-Gni *and $\mathcal{I}^- = \{j \in \mathcal{I} : c_j - \boldsymbol{a}_j^T\boldsymbol{\pi}^\star < 0\}$. Then*

1. *$\mathcal{I}^- = \emptyset$ if $z^\star_{\text{C-Gni}} \ge 0$.*
2. *$\mathcal{I}^- \supset \{j \in \mathcal{I} : d^\star_j > 0\}\ (\neq \emptyset)$ if $z^\star_{\text{C-Gni}} < 0$.*

Proof. Recall that $w_j > 0, \forall j \in \mathcal{L}$. Since λ is maximized in Cd-Gni, $\lambda^\star = z^\star_{\text{C-Gni}}$. The first set of constraints of Cd-Gni thus implies that $c_j - \boldsymbol{a}_j^T \boldsymbol{\pi}^\star \geq w_j z^\star_{\text{C-Gni}}$, $\forall j \in \mathcal{I}$. It follows that $\mathcal{I}^- = \emptyset$ if $z^\star_{\text{C-Gni}} \geq 0$.

Assume that $z^\star_{\text{C-Gni}} < 0$. For $j \in \mathcal{I}$ such that $d^*_j > 0$, the complementary slackness ensures that $c_j - \boldsymbol{a}_j^T \boldsymbol{\pi} = w_j z^\star_{\text{C-Gni}} < 0$, so $j \in \mathcal{I}^-$. As a consequence, $\{j \in \mathcal{I} : d^\star_i > 0\} \subset \mathcal{I}^-$. □

The optimal solution of Cd-Gni provides a dual solution of P, $\boldsymbol{\pi}^\star$, that satisfies the complementarity conditions $x_j \cdot (c_j - \boldsymbol{a}_j^T \boldsymbol{\pi}) = 0$, $j = 1, \ldots, n$, and maximizes the minimum normalized reduced cost $(c_j - \boldsymbol{a}_j^T \boldsymbol{\pi})/w_j$, $j = 1, \ldots, n$. Proposition 5 thus suggests a heuristic augmentation strategy that selects the variables that will potentially lead to the greatest improvements in the objective value. We initialize the columns of Red with those indexed by $\mathcal{P}_0 \cup C_0$ at step 7, and we enlarge the problem by including the variables of $\mathcal{I}^-$ at each augmentation phase. Proposition 5 then ensures that every incompatible variable taking a positive value in the improving direction $\boldsymbol{d}^\star$ is included in Red.

One last important issue of this augmentation phase is the update of the rows of Red. Since $\mathcal{I}^-$ may contain a large number of variables, the restriction of P to the variables indexed by $\mathcal{P} \cup C \cup \mathcal{I}^-$ does not necessarily have redundant constraints. The operations that would be necessary to identify the redundant rows are thus skipped, and all the rows of P are included in Red.

3.2.2. Case 2: the reduced problem has been augmented since the last reduction

Since every row is added to the reduced problem at the end of an augmentation phase (step 27), $|\mathcal{B}| = m$. The test performed at step 14 thus concludes that Gni will be used to select the variables of $\mathcal{I}$ that should be included in Red.

To explain this alternative in Algorithm 2, we denote by $(\mathcal{P}, C, \mathcal{I})$ the partition computed at step 13. Nothing was done during the previous augmentation phase(s) to ensure that the columns compatible with $\mathcal{P}$ were included in Red. If the columns indexed by C are not included in Red, solving Red to optimality does not necessarily imply that $z^\star_{\text{R-Gni}} = 0$, so Proposition 4 does not hold. As a consequence, the complete direction search problem Gni is solved instead of C-Gni. After we replace C-Gni with Gni, Proposition 4 holds and the remainder of the augmentation is unchanged.

Since C-Gni and Gni play the same role in Algorithm 2, they are both referred to as the complementary problem. This simplifies the presentation and emphasizes the correspondence with the complementary problem of IPS.

3.3. Warm-starting the complementary problem

As in IPS, Red is solved with the primal simplex, and C-Gni is solved with the dual simplex. An essential point for the efficiency of the overall algorithm is a good dual feasible basic solution to warm-start C-Gni (or Gni). We thus describe how such a basis is computed at steps 16 and 19 of Algorithm 2. In the following discussion, we extend the term *basis* to the set of indices $\mathcal{B}$ when the corresponding set of columns $\boldsymbol{A}_{.\mathcal{B}}$ is a basis.

Proposition 6. *Let $\mathcal{B}$ be a basis of* P, *$\boldsymbol{x}$ and $\bar{\boldsymbol{c}}$ the corresponding basic solution and reduced cost vector, and $(\mathcal{P}, C, \mathcal{I})$ the partition corresponding to $\boldsymbol{x}$.*

1. *Let $b = \underset{j \in \mathcal{L}}{\text{argmin}}\{\frac{\bar{c}_j}{w_j}\}$; then $\mathcal{B} \cup \{b\}$ is a dual feasible basis of* Gni *whose objective value is $z_{\text{Gni}} = \min_{j \in \mathcal{L}}\{\frac{\bar{c}_j}{w_j}\}$.*
2. *Let $b_I = \underset{j \in \mathcal{I}}{\text{argmin}}\{\frac{\bar{c}_j}{w_j}\}$; then $\mathcal{B} \cup \{b_I\}$ is a dual feasible basis of* C-Gni *whose objective value is $z_{\text{C-Gni}} = \min_{j \in \mathcal{I}}\{\frac{\bar{c}_j}{w_j}\}$.*

Proof. A proof of the second point is given in [17]; it can easily be adapted to show the first point. □

At step 16 of Algorithm 2, Red has m rows, so its optimal basis is also a basis of P, hence Proposition 6 shows that $\mathcal{B} \cup \{b\}$ can be used to warm-start Gni. If Red contains $|\mathcal{P}_0| = p_0$ rows, its optimal basis, $\mathcal{B}_0$, must be completed with $m - p_0$ columns to form a a basis of P. In IPS, $\mathcal{B}_0$ is completed with $m - p_0$ vectors of the canonical basis of $\mathbb{R}^m$ to form a basis $\mathcal{B}_1$ of P. This corresponds to adding slack variables to P and enforcing zero lower and upper bounds. These slack variables are systematically added by CPLEX, so no additional operation is required in our implementation. The dual feasible basis $\mathcal{B}_1^D = \mathcal{B}_1 \cup \{b_I\}$ is then a valid candidate to warm-start C-Gni.

The basis $\mathcal{B}_1^D$ represents a neutral but poor initial guess for C-Gni since it reflects only the dual information gathered when solving Red. In particular, the dual solution does not take into account the rows indexed by $\overline{\mathcal{P}}$. Another completion is thus considered.

Proposition 7. *Let $\mathcal{B}$ be a feasible basis of* P *and $(\mathcal{P}, C, \mathcal{I})$ be the partition associated with the corresponding basic solution. Let $\boldsymbol{A}_{.\mathcal{B}_0}$ be a set of independent columns spanning $\boldsymbol{A}_{.\mathcal{P}}$, and let $\bar{\mathcal{P}} = \mathcal{B} \setminus \mathcal{P}$. Then $\boldsymbol{A}_{.\mathcal{B}_0 \cup \bar{\mathcal{P}}}$ is a basis of $\mathbb{R}^m$. Moreover, if $\mathcal{B}_0$ is a feasible basis of P_{C_0}, then $\mathcal{B}_0 \cup \bar{\mathcal{P}}$ is a feasible basis of* P.

Proof. $\boldsymbol{A}_{.\mathcal{B}}$ is a basis of $\mathbb{R}^m$, so $\mathrm{Span}(\boldsymbol{A}_{.\mathcal{P}}) \oplus \mathrm{Span}(\boldsymbol{A}_{.\bar{\mathcal{P}}}) = \mathbb{R}^m$, which implies that $\boldsymbol{A}_{.\mathcal{B}_0 \cup \bar{\mathcal{P}}}$ is a basis of $\mathbb{R}^m$.

If $\mathcal{B}_0$ is a feasible basis of P_{C_0}, then $\boldsymbol{b} = \boldsymbol{A}_{.\mathcal{B}_0} \boldsymbol{x}_{\mathcal{B}_0}$ with $\boldsymbol{x}_{\mathcal{B}_0} \geq 0$. As a consequence, any completion of $\mathcal{B}_0$ is a feasible basis of P. □

Since Algorithm 2 starts with a basic feasible solution, a complete basis $\mathcal{B}$ of P is always known. At step 19, a completion of $\mathcal{B}$ based on Proposition 7 is thus possible. Let $\mathcal{B}_2$ be this basis of P, and $\mathcal{B}_2^D = \mathcal{B}_2 \cup \{b_I\}$. Then $\mathcal{B}_1^D$ and $\mathcal{B}_2^D$ are two valid candidates to warm-start C-Gni. Since Cd-Gni is a maximization problem, the basic solution with the higher objective value is chosen.

3.4. Maintaining a feasible basis

As for the complementary problem, it is important to start solving Red with a basis corresponding to the current solution. We must therefore find a basis corresponding to the updated solution after the augmentation (step 28).

Proposition 8. *Assume that* P *is bounded and that the current solution $\boldsymbol{x}$ at step 13 is not optimal. Let $\mathcal{B}^\star$ and $\boldsymbol{d}^\star$ be the optimal basis and solution of the complementary problem, and $b \in \{j \in \mathcal{P} : x_j + \rho^{\max} \boldsymbol{d}_j^\star = 0\}$. Then $\mathcal{B} = \mathcal{B}^\star \setminus \{b\}$ is a feasible basis of* P *and the corresponding basic solution is $\boldsymbol{x} + \rho^{\max} \boldsymbol{d}^\star$.*

Proof. According to Proposition 1, $\boldsymbol{x}$ is not optimal implies that $\boldsymbol{d}^\star \neq \boldsymbol{0}$ and P bounded implies that $\rho^{\max} < +\infty$, which shows that $\{j \in \mathcal{P} : x_j + \rho^{\max} \boldsymbol{d}_j^\star = 0\}$ is not empty. Next, $\boldsymbol{d}^\star$ is the basic solution of Gni associated with $\mathcal{B}^\star$, so $\boldsymbol{A}_{.\mathcal{B}^\star} \boldsymbol{d}^\star = 0$, and $\boldsymbol{A}_{.\mathcal{B}} \boldsymbol{d}_{\mathcal{B}}^\star = -d_b^\star \boldsymbol{a}_b$. Since $d_b \neq 0$, $\boldsymbol{a}_b \in \mathrm{Span}(\boldsymbol{A}_{.\mathcal{B}})$ and the rank of $\boldsymbol{A}_{.\mathcal{B}}$ and $\boldsymbol{A}_{.\mathcal{B}^\star}$ are the same. Moreover, $\boldsymbol{A}_{.\mathcal{B}^\star}$ is obtained by removing one row from the basic matrix of Gni, whose rank is equal to $m + 1$, so the ranks of $\boldsymbol{A}_{.\mathcal{B}^\star}$ and $\boldsymbol{A}_{.\mathcal{B}}$ are equal to m. As a consequence, $\mathcal{B}$ is a basis of P.

To see that $\boldsymbol{x} + \rho^{\max} \boldsymbol{d}^\star$ is the corresponding basic solution, first recall that the initial basis of Gni includes $\mathcal{P}$ and that the variables of $\boldsymbol{d}_{\mathcal{P}}$ are unbounded. These variables cannot be chosen as the exiting variable in a dual pivot and thus remain in the basis until optimality, so $\mathcal{B}^\star \supset \{j : x_j > 0\} \cup \{j : d_j^\star \neq 0\}$. As a consequence, $\boldsymbol{A}_{.\mathcal{B}} \left(\boldsymbol{x} + \rho^{\max} \boldsymbol{d}_{\mathcal{B}}^\star \right) = \boldsymbol{b} + \boldsymbol{0}$ and $\mathcal{B} \supset \{j : x_j + \rho^{\max} d_j^\star > 0\}$, which completes the proof. □

With this last proposition, it is finally possible to address the convergence of the overall algorithm.

Theorem 2. *Let* P *be a feasible LP. Assuming that the primal and dual simplex are finite, Algorithm 2 is finite and returns either an optimal solution of* P *if it is bounded, or an optimal ray if it is unbounded.*

Proof. If Algorithm 2 is finite, it either ends because Red is found to be unbounded, or because the optimal solution of Gni satisfies either $z_{\mathrm{Gni}}^\star = 0$ or $\boldsymbol{d}^\star \geq \boldsymbol{0}$. If an optimal ray of Red is found, it can be completed with zeros to form an optimal ray of P. Otherwise, let $\boldsymbol{x}^\star$ be the result of the algorithm. If $z_{\mathrm{Gni}}^\star = 0$ or $\boldsymbol{d}^\star \geq \boldsymbol{0}$, Proposition 1 ensures that $\boldsymbol{x}^\star$ is respectively an optimal solution or an optimal ray of P.

To prove that the algorithm is finite, we first notice that each step is finite if the primal and dual simplex are assumed to be finite. For the same reason, the loop covering step 4 to step 12 eventually end with an optimal solution of Red, so the algorithm performs finite major iterations starting at step 4 and ending at step 27. Let $\boldsymbol{x}$ be the current solution before solving C-Gni or Gni; Proposition 4 shows that if $\boldsymbol{x}$ is not optimal, then $\boldsymbol{d}^\star$ is an improving feasible direction. Moreover, Proposition 8 ensures that $\boldsymbol{x} + \rho^{\max} \cdot \boldsymbol{d}^\star$ is a basic solution of P. As a consequence, each major iteration ends with a basic feasible solution providing a strict improvement in the objective value. Since the number of basic solutions is finite, so is Algorithm 2. □

4. Computational tests

In this section, we evaluate IPSO by comparing it to the primal simplex of CPLEX 12.4. The pricing criterion is systematically set to an approximate version of the steepest edge described by Goldfarb and Reid [12] (CPX_PARAM_PRIIND=3), because it has similar performance to that of the automatic default pricing of CPLEX

but allows for a consistent analysis of the results (see [17]). The other CPLEX parameters are set to their default values. We finally compare the performance of IPSO with that of IPS and with that reported by Towhidi et al. [26] for the primal simplex with the positive edge pricing criterion.

The tests are all performed on an OpenSuse operating system with an Intel(R) Core(TM) i7-3770 CPU @ 3.40 GHz processor, and all the LPs, including Red, C-Gni, and Cr-Gni, are solved with the simplex of CPLEX. The initial basic feasible solutions used by every algorithm are obtained by the primal simplex phase I of CPLEX. We set the time limit to 10 hours for every tested algorithm.

Unless otherwise specified, CPLEX refers to the primal simplex of the LP solver in the rest of the article.

4.1. Implementation improvements

In their analysis of the performance of IPS, Omer et al. [17] emphasize that the algorithm was designed with some particular classes of highly degenerate problems in mind. It would be unrealistic to expect that it would achieve the impressive performance described in [6, 24] on every LP. Omer et al. [17] identify three major reasons that IPS may not perform well on some instances. The first two relate to degeneracy. If only a few simplex pivots are degenerate ($< 30\%$), then interrupting the primal simplex may be counterproductive. Otherwise, if only a few variables are degenerate ($< 20\%$), then reducing P is unlikely to lead to a faster solution of the problem. Moreover, after a reduction of Red, the problem contains only the positive and compatible variables. If Red contains only a few compatible variables, it is again likely that the whole process of dynamic reduction will not be advantageous. Referring to [17], IPS does not perform well when $|C_0| \leq 0.5\,|\mathcal{P}_0|$. In IPS, identifying C is time-consuming, but this is not the case if the positive edge criterion is used. As a consequence, one important contribution of IPSO is that we can test its potential for improvement in advance.

In practice, Algorithm 3 is run before Algorithm 2 to take into consideration the conclusions of [17]. This algorithm iteratively tests the potential of IPSO and perform simplex pivots on P until the potential of IPSO is shown or optimality is reached. A few pivots are done before the first time that the numbers of degenerate and compatible variables are tested to estimate the percentage of degenerate pivots. The maximum number of pivots M is then increased after each interruption of the simplex to limit the number of these interruptions when the tests are unsuccessful.

Algorithm 3: Testing the potential of IPSO

```
1  x^0 ← a basic feasible solution;
2  M ← 1000; d ← 0; optimal← false; potential← false;
3  while x^0 is not optimal and potential = false do
4      while d < 0.3 × M and optimal = false do
5          Perfom M simplex pivots on P;
6          d ← number of degenerate pivots;
7          x^0 ← the current solution of P;
8          if x^0 is optimal or Red is unbounded then
9              optimal← true;
10     p ← |{j : x_j^0 > 0}|;
11     if p ≤ 0.8m then
12         Use the positive edge criterion to build the partition (P_0, C_0, I_0) corresponding to x^0;
13         if |C_0| ≥ 0.5 |P_0| then
14             potential← true;
15     M ← M + m;
```

If Algorithm 3 reaches step 14, IPSO will potentially improve the primal simplex, so the current solution $\boldsymbol{x}^0$ is used to initialize $\boldsymbol{x}$ at step 2 of Algorithm 2. Otherwise, the complete solution of the problem is done in Algorithm 3 with primal simplex pivots on P. In the latter case, the difference with an ordinary execution of the primal simplex is that the algorithm is interrupted a limited number of times to make unsuccessful tests. The only costly operation in the tests of Algorithm 3 is the identification of the compatible variables when building the partition $(\mathcal{P}_0, C_0, \mathcal{I}_0)$. As is stated in Section 3.1, this requires a similar number of operations as computing the reduced costs, so the complexity of

the tests is the same as that of a simplex pivot. For a very small computational cost, Algorithm 3 may thus prevent from starting the dynamic reduction of IPSO on instances that should be solved more efficiently with the primal simplex.

Another improvement ensures that IPSO performs well on highly degenerate LPs with only a few compatible variables. When the number of compatible variables is not sufficient to start IPSO, but the simplex performs a large number of degenerate pivots, it is interrupted every time it remains stuck on the same vertex for more than some large number of pivots. No reduction is done, but GNI is then solved to find a feasible improving direction. In our tests, this process is triggered when more than 95% degenerate pivots are performed overall and the objective value stalls for more than 1000 pivots.

4.2. Description of the benchmark

The benchmark includes the VCS, UBFA, and UFL instances used in [17], and 45 instances of Mittelmann's benchmark[3]. For Mittelmann's benchmark, we exclude only two instances that take less than one second to solve. As in [17], we added the dual formulation of four LPs (neos1, neos2, neos3, rlfprim) whose number of constraints is much larger than the number of variables. The characteristics of the benchmark appear in Table 1, which includes the dimensions ($n \times m$) and the density (ρ_A) of $\boldsymbol{A}$, and the main statistics for the phase II of the primal simplex of CPLEX, i.e., the number of pivots (piv), the percentage of degenerate pivots (degen piv), and the computational time (cpu). The CPLEX statistics result from *two separate optimization runs for each model* : the number of pivots and the total runtime are obtained with the default options, but the perturbations had to be disabled to compute the fraction of degenerate pivots. The column "phase I" indicates how much time is spent in phase I to find an initial feasible solution. Since the time and pivots spent in this phase I are common to every algorithm, they are excluded from the subsequent comparisons.

4.3. Comparison with CPLEX

In our first evaluation of IPSO we present detailed statistics on its execution and compare it with the primal simplex of CPLEX. The tests described in Algorithm 3 are performed before starting the dynamic reduction. The results of the instances solved with the dynamic reduction are reported in Table 2. The instances that are not degenerate enough or have too few compatible variables appear in Table 3. In both tables, we do not report the time spent in testing the potential of IPSO, because it is negligible compared to the solution times of IPSO and CPLEX.

We first focus on Table 2, where "aug" is the number of augmentation phases, and $\frac{m-p}{m}$ and $\frac{|C|}{|\mathcal{P}|}$ are the average percentages of degenerate and compatible variables. The next four columns detail the time spent and the numbers of pivots performed when solving the reduced and the complementary problems. The remaining runtime is that spent in identifying the compatible variables and the redundant constraints of P_C; we do not report it in Table 2, because it is negligible when compared to the solution time of the reduced and complementary problems. The last two columns contain the ratios of the total number of pivots and runtime of CPLEX relative to those of IPSO, thus indicating improvement factors. The cpu improvement column shows that IPSO significantly ($\geq 30\%$) outperforms CPLEX on 33 of 38 instances while CPLEX never outperforms IPSO, which means that Algorithm 3 successfully filters out the unfavorable cases. Moreover, IPSO divided the computational time by a factor larger than two on all the VCS and UBFA instances and on nine Mittelmann instances. Most of the computational time was spent in the reduced problem, which is consistent with the fact that only one complementary problem is solved at each augmentation phase. Finally, the six instances buildingen, neos1, neos2, neos3', rlfprim', and stat96v1 have very low $\frac{|C|}{|\mathcal{P}|}$, so P is never reduced during the solution with IPSO. The improvement in runtime is achieved by solving one or more complementary problems to find improving feasible directions, as described in Section 4.1.

Table 3 details the degeneracy and compatibility values when the tests in Algorithm 3 are not successful. The instances are organized according to the test that fails, and we report the highest observed values. IPS and IPSO are designed to take advantage of degeneracy, so it seems logical that the first twelve instances should not be tested. On the other hand, when solving the last ten instances, the primal simplex struggles with degeneracy, and the number of degenerate variables would allow for an advantageous reduction of P. IPSO was thus executed without testing the number of compatible variables to evaluate the relevance of the chosen threshold. For eight instances out of ten, IPSO is outperformed by CPLEX although the improvement factor is significant in only five cases. This shows that

[3] These instances are available online: `http://plato.asu.edu/ftp/lpcom.html` (last visited on January 14th, 2015).

Table 1: Characteristics of the benchmark

	Size of P			CPLEX solve			
Instance	m: rows	n: cols	ρ_A	phase I (s)	piv	cpu (s)	degen piv
VCS1	2084	10343	1.5E-2	5.19	21091	20.30	70.2%
VCS2	2084	10150	1.6E-2	6.91	22980	21.95	72.2%
VCS3	2085	26350	1.7E-2	6.11	42341	93.07	47.9%
VCS4	1200	133572	1.7E-2	4.33	12479	68.43	66.3%
VCS5	1600	570983	1.2E-2	17.35	25849	607.08	66.9%
UBFA1	5182	23650	2.5E-3	21.68	11129	8.92	63.9%
UBFA2	5182	23990	2.5E-3	28.88	15566	13.09	58.3%
UBFA3	5182	24282	2.5E-3	30.04	67792	93.72	68.5%
UBFA4	5182	24517	2.5E-3	27.86	142550	255.36	73.2%
UBFA5	5182	24875	2.5E-3	31.91	53591	73.42	85.8%
UWL2	7965	7965	2.4E-4	0.02	20327	5.31	40.9%
UWL3	10440	10440	1.9E-4	0.02	36072	8.99	36.8%
UWL4	15476	30452	1.3E-4	0.03	54069	17.57	61.3%
UWL5	20534	40568	9.6E-5	0.04	60587	23.72	71.4%
UWL6	25931	51462	7.7E-5	0.05	58382	23.21	84.3%
buildingen	277594	154978	9.2E-6	0.75	169070	282.53	73.2%
cont1	160792	40398	2.0E-5	235.54	2227	151.12	91.3%
cont11	160792	80396	1.6E-5	236.15	90026	4188.87	28.6%
cont4	160792	40398	2.0E-5	88.89	830	68.63	99.0%
dano3mip	3202	15851	1.6E-3	0.15	13785	5.22	60.1%
dbic1	43200	226317	1.1E-4	0.38	28371	12.44	88.3%
dfl001	6071	12230	4.8E-4	1.88	21895	8.77	70.8%
ds-big	1042	174997	2.5E-2	2.50	24818	280.85	46.0%
fome12	24284	48920	1.2E-4	2.97	78964	66.61	66.6%
fome13	48568	97840	6.0E-5	6.89	171371	240.75	73.0%
gen4	1537	4297	1.6E-2	10.21	2072	3.24	78.3%
ken-18	105127	154699	2.2E-5	0.80	66767	21.59	32.5%
L1-simx	986069	428032	1.0E-5	4.02	89021	3852.64	95.9%
l30	2701	16281	1.2E-3	0.02	123704	136.84	98.0%
Linf_520c	93326	103505	6.2E-5	50.99	23319	138.95	45.1%
lp22	2958	16392	1.4E-3	3.18	36250	20.38	80.0%
mod2	34774	66409	8.7E-5	5.16	38023	43.79	12.1%
neos	479119	515905	6.2E-9	24.23	12721	17.16	53.8%
neos1	131581	1892	3.4E-5	16.36	16588	37.33	99.9%
neos1'	1892	131582	1.9E-3	0.18	33580	111.98	99.0 %
neos2	132568	1560	3.9E-5	16.39	93410	398.11	100.0%
neos2'	1560	132568	2.6E-3	0.29	39460	149.87	99.2 %
neos3	512209	6624	4.5E-4	0.82	>527758	>36000	96.2%
neos3'	6624	512209	4.5E-4	0.62	61140	939.58	1.7%
ns1687037	50622	43749	3.5E-4	2.01	140664	4506.89	35.14%
ns1688926	32768	41163	1.3E-3	0.40	18969	90.81	30.9%
nsct2	23003	37563	8.1E-4	0.14	6847	2.69	87.3%
nug08-3rd	19728	20448	2.5E-4	1083.13	247897	6724.86	99.9%
nug15	6330	22275	6.7E-4	46.03	92087	257.49	55.6%
pds-040	66844	217531	3.2E-5	1.48	41587	34.41	87.6%
pds-100	156243	514577	1.4E-5	6.99	174460	530.01	91.6%
qap12	3192	8856	1.4E-3	7.32	20727	17.21	32.8%
qap15	6330	22275	6.7E-4	82.70	108918	339.23	60.6%
rail2586	2586	923269	3.4E-3	2.81	45855	1248.37	36.0%
rail4284	4284	1096890	2.4E-3	4.21	88617	2520.75	52.4%
rlfprim	58866	62716	8.7E-5	0.11	15214	8.86	99.6%
rlfprim'	8052	74970	4.7E-4	0.13	3224	2.31	100.0%
stat96v1	5995	197472	5.0E-4	0.36	18845	84.89	56.5%
stat96v4	3173	63076	2.5E-3	29.47	43352	113.74	43.0%
storm_1000	528185	1377310	4.8E-6	9.56	221870	1219.95	58.9%
storm_125	66185	172431	3.8E-5	0.40	27810	9.30	50.2%
stp3dlp	159488	336283	1.5E-5	817.07	260848	1679.04	97.7%
watson_1	201155	386992	1.4E-5	16.97	66106	84.27	79.5%
watson_2	352013	677224	7.7E-6	20.91	124865	290.47	69.5%
world	34506	67147	8.6E-5	2.65	44387	56.85	13.7%

Table 2: Comparison of IPSO with CPLEX

		potential		solve GNI		solve RED		total	improvement	
	aug	$\frac{m-p}{m}$	$\frac{\lvert C\rvert}{\lvert\mathcal{P}\rvert}$	piv	cpu	piv	cpu	cpu	piv	cpu
VCS1	9	43%	437%	339	0.35	21616	11.85	12.6	0.98	1.61
VCS2	10	45%	447%	556	0.45	25180	13.22	14.12	0.91	1.55
VCS3	9	60%	154%	6239	4.46	27236	18.87	23.54	1.26	3.95
VCS4	1	43%	17252%	91	0.16	8494	25.37	25.63	1.45	2.67
VCS5	3	53%	52541%	213	3.21	21133	239.81	243.44	1.21	2.49
Average									**1.15**	**2.31**
UBFA1	4	73%	302%	1736	1.14	15362	6.52	7.78	0.72	1.15
UBFA2	5	74%	312%	1817	1.27	17573	7.43	8.88	0.89	1.47
UBFA3	4	75%	331%	3483	1.77	18800	6.98	8.81	3.04	10.64
UBFA4	6	76%	280%	8873	4.62	30378	12.13	16.82	3.63	15.18
UBFA5	3	73%	324%	1471	0.78	16799	9.62	10.43	2.93	7.04
Average									**1.83**	**4.54**
UWL2	4	52%	101%	1	0.01	21156	4.46	4.53	0.96	1.17
UWL3	4	66%	67%	8	0.01	24578	5.94	6.03	1.47	1.49
UWL4	3	74%	75%	3	0.08	40051	9.70	9.79	1.35	1.79
UWL5	3	79%	69%	130	0.14	51901	13.38	13.55	1.16	1.75
UWL6	3	81%	81%	19	0.13	42476	13.25	13.41	1.37	1.73
Average									**1.25**	**1.57**
buildingen	2	77%	20%	11860	7.66	147889	161.97	170.54	1.14	1.66
dano3mip	4	33%	419%	21	0.07	15251	3.71	3.81	0.90	1.37
dbic1	3	89%	105%	13	0.66	17886	5.29	6.06	1.59	2.05
ds-big	4	35%	32216%	787	4.73	33015	258.04	263.83	0.75	1.06
gen4	1	68%	0%	0	0.02	1001	1.57	1.6	2.07	2.03
l30	4	25%	176%	4787	2.12	19698	10.49	12.64	5.05	10.83
lp22	4	28%	389%	2120	0.89	24954	13.43	14.37	1.34	1.42
L1_simx	16	96%	100%	17221	988.8	79374	1824.58	2823.54	1.12	1.36
neos1	3	50%	0%	3073	8.89	3282	8.03	17.18	2.61	2.17
neos1'	3	86%	4942%	94	0.24	2374	0.60	0.88	13.61	127.83
neos2	5	64%	1%	12466	42.47	6183	15.65	58.53	5.01	6.80
neos2'	3	80%	4110%	1384	2.03	13154	5.31	7.38	2.71	20.32
neos3	7	84%	11%	305399	4608.34	85123	4651.2	9262.82	>6.20	> 3.80
neos3'	2	38%	50%	47	0.39	6101	73.97	74.97	10.02	12.53
nug08-3rd	4	100%	18435%	106887	731.64	21735	590.62	1339.39	11.41	5.02
rail2586	3	26%	16489%	314	5.66	52224	555.94	561.99	0.87	2.22
rail4284	3	26%	6461%	1004	19.82	79452	713.97	734.30	1.10	3.43
rlfprim'	1	48%	3%	2970	1.24	101	0.02	1.28	1.05	1.81
stat96v1	1	21%	13%	5223	16.00	13446	57.83	74.05	1.01	1.15
storm_1000	3	56%	112%	124	7.60	179106	704.72	714.08	1.24	1.71
storm_125	3	56%	111%	1	0.48	29325	8.71	9.28	0.95	1.00
watson_1	3	83%	62%	18	1.46	30152	8.99	10.69	2.19	7.89
watson_2	3	80%	63%	14	2.24	40879	61.70	64.38	3.05	4.51
Average									**2.07**	**3.40**

Table 3: Instances with low potential improvement

	degen piv	$\frac{m-p}{m}$	$\frac{\|C\|}{\|\mathcal{P}\|}$	cpu improve	
mod2	12.5%				← degen piv < 30%
qap12	21.4%				
world	13.7%				
cont1	58.1%	0.0%			← $\frac{m-p}{m}$ < 20%
cont11	34.7%	0.6%			
cont4	99.0%	0.0%			
neos	53.6%	4.1%			
ns1687037	35.1%	4.5%			
nug15	23.8%	13.8%			
qap15	53.3%	12.2%			
rlfprim	99.5%	17.8%			
stat96v4	41.2%	1.0%			
dfl001	43.6%	45.2%	20.1%	0.89	← $\frac{\|C\|}{\|\mathcal{P}\|}$ < 50%
fome12	73.1%	31.4%	39.9%	0.68	
fome13	77.5%	40.9%	45.2%	0.81	
ken-18	33.4%	29.7%	20.3%	0.67	
Linf-520c	54.3%	27%	41%	0.88	
ns1688926	41.5%	44.4%	5.9%	1.30	
nsct2	87.8%	49.0%	3.0%	1.83	
pds-040	87%	82%	26%	0.35	
pds-100	91%	88%	34%	0.60	
stp3dlp	98.5%	94.7%	19.7%	0.65	

the decomposition based on compatibility does not allow us to take advantage of degeneracy for every degenerate LP (only 21 of 34 degenerate Mittelmann instances), but it is possible to test the potential efficiency of IPSO before starting the dynamic reduction. Moreover, this test is fast, since the positive edge criterion allows us to find the number of compatible variables within a computational time that compares to that of a simplex pivot. As a consequence, the algorithm could be used as an adaptive strategy integrated in an efficient primal simplex.

For a more general view on the available LP solvers, we compared IPSO and the primal simplex of CPLEX with the barrier and dual simplex algorithms of CPLEX. These comparisons focus on the solution times from scratch. The results show that the dual simplex and the barrier algorithm respectively outperform IPSO by an average 1.55 and 2.15 factor, whereas they outperform the primal simplex by an average 2.90 and 4.10 factor. Even though IPSO is clearly more competitive than the primal simplex, the improvement is not sufficient to make it the most efficient algorithm when solving an LP from scratch. Despite this, IPSO provides the best solution time for 10% of the instances, whereas the primal simplex outperforms the other three algorithms for only one instance. IPSO may then be useful as one of the concurrent methods used in the default behavior of CPLEX. Moreover, as highlighted in the introduction, the main benefit of the primal simplex is that it can be used to efficiently optimize an LP when a good feasible solution is available, so IPSO should be the best option in a large number of such situations.

4.4. Comparison with IPS and positive edge

In this section we compare IPSO with the latest version of IPS described in [17] and with the results reported by Towhidi et al. in [26] for the primal simplex of CLP including the positive edge pricing criterion (PE).

Table 4 gives the computational time spent identifying the compatible variables ("$\mathcal{PCI}$") and solving the reduced problems ("Red") and the complementary problem ("Cr-Gni") during the execution of IPS, and total runtime of IPS ("total"). The last three columns give the cpu improvement of IPS, IPSO, and PE (when available in [26]). The cpu improvement of IPS and IPSO are the ratios of the runtime of the primal simplex of CPLEX relative to those of IPS and IPSO, respectively. The cpu improvement of PE is the ratio of the runtime of CLP's primal simplex relative to that of PE. In our tests IPS is started from the same phase I feasible solution as IPSO, whereas PE uses a different solution generated by CLP in [26]. Only the Mittelmann instances with sufficient degeneracy are included, and they are divided

into two different groups depending on whether or not the compatibility test fails ($\frac{|\mathcal{C}|}{|\mathcal{P}|} < 50\%$). The instances are then organized by increasing number of rows.

The improvement factors show that IPS and IPSO have similar performance on the VCS and UFL problems. IPS outperforms IPSO on the UBFA instances, but the improvement factors are still high. Focusing on the first group of Mittelmann instances, IPS matches the performance of IPSO in four of sixteen cases but is significantly outperformed in the other twelve cases. Moreover, IPS experiences serious difficulties on the largest instances, while there is no obvious correlation between the performance of IPSO and the size of the instance. To explain the difficulties of IPS, we observe that the identification of compatible variables alone takes more time than the complete solution by CPLEX for buildingen, dbic1, neos3, pds-100, storm_125, storm_1000, watson_1, and watson_2. Moreover, CR-GNI must be solved a large number of times during the augmentation phases to select a significant number of incompatible variables. Finally, IPS does not perform better on the instances that would fail the compatibility test, but this test is time-consuming if $\bar{\boldsymbol{A}}$ has to be computed.

Towhidi et al. [26] use the positive edge criterion as a pricing criterion to select the compatible variables. More precisely, if $j_C = \text{argmin}\{\bar{c}_j : j \in C\}$ and $j_I = \text{argmin}\{\bar{c}_j : j \in I\}$, the entering variable is x_{j_C} if $\bar{c}_{j_C} < \psi \bar{c}_{j_I}$ and x_{j_I} otherwise ($0 < \psi < 1$). They thus use the concept of compatibility to reduce the number of degenerate steps, but they do not take advantage of degeneracy by removing constraints from the LP. The comparison with PE is presented simply to indicate global trends, because the available data is partial and, more importantly, because the improvement factors of PE refer to CLP. Nevertheless, if we focus on the instances that satisfy the compatibility test, we see that IPSO's improvement factors range from two to six for rail4284, neos1, neos2, and watson-2 whereas PE fails to improve the computational time of CLP. For rail4284 and watson-2, the success of IPSO is due to the reduction of P, but for neos1 and neos2, solving GNI to find improving feasible directions makes the difference. On the other hand, PE clearly outperforms IPSO on all the instances that failed the compatibility test. In these cases, the advantage of PE is its flexibility. In contrast with IPSO, it does not exclusively focus on the compatible variables and does not need to rely on a heuristic augmentation phase. PE selects a compatible variable when its reduced cost is negative enough and an incompatible variable otherwise.

5. Conclusion

The improved primal simplex (IPS) is a dynamic constraint reduction algorithm that takes advantage of degeneracy in LPs. Among the pitfalls that remain to be addressed are the costly matrix product performed when identifying the compatible variables, and the numerous solutions of the complementary problem at each augmentation phase. In this work, we introduce two major modifications to overcome these pitfalls. The first is the use of the positive edge criterion [23] as a fast stochastic compatibility test. This test allows for a fast reduction of the LP during the solution process, and it also offers an opportunity to rapidly test the potential performance of our method prior to starting it. The second is the revision of the whole augmentation phase to focus on a complementary problem involving the original coefficients of the matrix, which is solved only once per phase. We also describe how to build good dual and primal feasible bases to warm-start respectively the complementary and the reduced problems. The resulting algorithm, referred to as IPSO, is evaluated on a benchmark including 45 Mittelmann instances. This evaluation shows that the preemptive test efficiently identifies the instances that have a limited potential for improvement with IPSO. When this test is successful, IPSO outperforms the primal simplex of CPLEX on all but five instances, and improvement factors greater than two are found in 20 of 38 cases. IPSO also outperforms IPS on all but two Mittelmann instances. To be specific, IPS has difficulties on the largest Mittelmann instances, whereas the improvement factor of IPSO is not correlated to the size of the instance. Finally, the comparison with the PE that Towhidi et al. [26] implemented in the primal simplex of COIN-OR's CLP offers important insight into future improvements of the algorithm. The reduction of the number of constraints is clearly an advantage of IPSO when there is a large number of compatible variables. On the other hand, the possibility of considering incompatible variables when the compatible variables do not have good reduced costs allows PE to perform well on every highly degenerate LP of Mittelmann's benchmark.

In future work, we will look into the possibility of a hybrid procedure combining IPSO and PE in COIN-OR's CLP. Although the primal simplex of CLP is not as fast as that of CPLEX, it has the great advantage of being an open code. The algorithm could then be implemented as an internal procedure, and the analysis would take advantage of the availability of the code. Another possibility is to extend the algorithm to the simplex phase I with a method similar to that of Pan [21]. It would also be interesting to investigate a dual version of IPSO. Pan [19] takes advantage of primal

Table 4: Comparison of IPSO with IPS [17] and PE [26]

	size of P		IPS: cpu time (s)				cpu improvement		
	m: rows	n: cols	$\mathcal{PCI}$	RED	CR-GNI	total	IPS	IPSO	PE
VCS1	2084	10343	0.54	6.19	0.69	7.53	2.70	1.61	–
VCS2	2084	10150	0.53	5.91	0.80	7.37	2.98	1.55	–
VCS3	2085	26350	1.64	6.41	7.14	15.72	5.92	3.95	–
VCS4	1200	133572	1.87	20.82	0.55	23.37	2.93	2.67	–
VCS5	1600	570983	13.11	146.13	63.27	226.42	2.68	2.49	–
UBFA1	5182	23650	1.06	2.75	2.09	6.08	1.47	1.15	–
UBF2	5182	23990	1.07	2.87	2.30	6.47	2.02	1.47	–
UBFA3	5182	24282	1.07	3.26	2.17	6.72	13.94	10.64	–
UBFA4	5182	24517	1.14	3.34	3.06	7.77	32.86	15.18	–
UBFA5	5182	24875	1.08	3.58	2.36	7.26	10.12	7.04	–
UWL2	7965	7965	0.70	2.16	0.81	3.79	1.40	1.17	–
UWL3	10440	10440	1.13	1.68	1.39	4.37	2.06	1.49	–
UWL4	15476	30452	2.93	3.37	2.42	9.00	1.95	1.79	–
UWL5	20534	40568	4.58	4.80	4.47	14.23	1.67	1.75	–
UWL6	25931	51462	8.08	3.65	6.26	18.44	1.26	1.73	–
ds-big	1042	174997	29.50	172.42	24.22	227.52	1.23	1.06	–
gen4	1537	4297	0.52	4.47	0.00	4.98	0.65	2.03	–
neos2dual	1560	134128	2.38	0.63	9.47	12.77	11.74	20.32	–
neos1dual	1892	133473	2.09	0.19	5.32	8.10	13.82	127.83	–
rail2586	2586	923269	85.51	246.60	275.60	619.64	2.01	2.22	–
l30	2701	16281	0.32	15.80	12.26	28.56	4.79	10.83	–
lp22	2958	16392	0.59	12.68	1.05	14.40	1.41	1.42	1.84
dano3	3202	15851	0.40	2.73	0.51	3.73	1.40	1.37	1.68
rail4284	4284	1096890	155.60	539.21	1336.12	2048.01	1.23	3.43	0.45
stat96v1	5995	197472	29.12	57.83	16.00	93.18	0.91	1.15	0.83
neos3’	6624	512209	122.36	312.91	35558.46	stopped	stopped	12.53	–
rlfdual	8052	74970	2.00	0.02	0.42	2.49	0.93	1.81	–
nug08-3rd	19728	20448	129.48	689.33	1463.33	2288.31	2.94	5.02	0.88
dbic1	43200	226317	51.24	6.96	535.17	593.42	0.02	2.05	2.15
neos1	131581	1892	150.76	478.19	5704.07	6333.47	0.06	2.17	1.05
neos2	132568	1560	829.40	4301.60	30878.50	stopped	stopped	6.80	1.03
storm_125	159488	336283	59.08	5.51	188.21	252.89	0.04	1.71	–
watson_1	201155	386992	579.45	3.87	2224.27	2807.69	0.03	7.89	–
buildingen	277594	154978	912.87	490.30	34527.22	stopped	stopped	1.66	–
watson_2	352013	677224	2522.88	58.43	17771.40	20487.62	0.01	4.51	1.81
neos3	512209	6624	1982.01	0.25	34017.99	stopped	stopped	>3.88	1.46
storm_1000	528185	1377310	8235.06	382.76	27764.94	stopped	stopped	1.00	–
L1_simx	986069	428032	3683.27	728.86	31964.68	stopped	stopped	1.36	–
dfl001	6071	12230	1.11	5.17	3.48	9.95	0.88	0.89	–
nsct2	23003	37563	3.39	0.27	14.06	17.74	2.47	1.83	–
fome12	24284	48920	16.80	55.98	111.42	187.39	0.36	0.68	1.52
ns1688926	32768	41163	5.92	26.85	14.00	46.77	0.44	1.30	2.91
fome13	48568	97840	24.02	230.09	192.58	446.81	0.54	0.81	2.54
pds-040	66844	217531	42.65	76.85	1048.54	1168.27	0.03	0.35	2.20
Linf-520c	93326	103505	98.81	2966.83	1758.12	4831.06	0.03	0.88	–
ken-18	105127	154699	34.76	62.05	112.13	209.05	0.02	0.67	–
pds-100	156243	514577	892.97	871.22	19945.92	21866.86	0.02	0.60	3.17
stp3dlp	159488	172431	200.01	92.49	3009.68	3302.36	0.03	0.65	–

degeneracy in a dual algorithm, but he does not treat dual degeneracy. Finally, another decomposition algorithm [11] reduces the size of the basis by giving special treatment to slack variables. A study of the interaction of IPSO with this decomposition could lead to additional improvements of the primal simplex.

References

References

[1] Benichou, M., Gauthier, J. M., Hentges, G., Ribiere, G., 1977. The efficient solution of large-scale linear programming problems: Some algorithmic techniques and computational results. Mathematical Programming 13 (1), 280–322.
[2] Bland, R. G., 1977. New finite pivoting rules for the simplex method. Mathematics of Operations Research 2 (2), 103–107.
[3] Charnes, A., 1952. Optimality and degeneracy in linear programming. Econometrica 20 (2), pp. 160–170.
[4] Dantzig, G. B., 1955. The general simplex method for minimizing a linear form under inequality constraints. Pacific Journal of Mathematics 5, 183–195.
[5] Davis, T. A., Duff, I. S., Jan. 1997. An unsymmetric-pattern multifrontal method for sparse LU factorization. SIAM J. Matrix Anal. Appl. 18 (1), 140–158.
[6] Elhallaoui, I., Metrane, A., Desaulniers, G., Soumis, F., 2011. An improved primal simplex algorithm for degenerate linear programs. INFORMS Journal on Computing 23 (4), 569–577.
[7] Elhallaoui, I., Metrane, A., Soumis, F., Desaulniers, G., 2010. Multi-phase dynamic constraint aggregation for set partitioning type problems. Mathematical Programming 123 (2), 345–370.
[8] Elhallaoui, I., Villeneuve, D., Soumis, F., Desaulniers, G., 2005. Dynamic aggregation of set-partitioning constraints in column generation. Operations Research 53 (4), 632–645.
[9] Gal, T., 1993. Selected bibliography on degeneracy. Annals of Operations Research 46-47 (1), 1–7.
[10] Gill, P. E., Murray, W., Saunders, M. A., Wright, M. H., 1989. A practical anti-cycling procedure for linearly constrained optimization. Mathematical Programming 45 (1-3), 437–474.
[11] Gleixner, A. M., 2012. Factorization and update of a reduced basis matrix for the revised simplex method. Tech. Rep. 12-36, ZIB, Takustr.7, 14195 Berlin.
[12] Goldfarb, D., Reid, J., 1977. A practicable steepest-edge simplex algorithm. Mathematical Programming 12, 361–371.
[13] Greenberg, H. J., 1978. Design and Implementation of Optimization Software. Sijthoff & Noordhoff, Ch. Pivot selection tactics, pp. 109–142.
[14] Harris, P. M. J., 1973. Pivot selection method of the devex LP code. Mathematical Programming 5, 1–28.
[15] Murtagh, B., Saunders, M., 1978. Large-scale linearly constrained optimization. Mathematical Programming 14 (1), 41–72.
[16] Murtagh, B. A., Saunders, M. A., 1983. Minos 5.0 user's guide. Tech. rep., DTIC Document.
[17] Omer, J., Rosat, S., Raymond, V., Soumis, F., 2014. Improved primal simplex: A more general theoretical framework and an extended experimental analysis. Submitted to INFORMS Journal on Computing.
[18] Pan, P.-Q., 1998. A basis-deficiency-allowing variation of the simplex method for linear programming. Computers & Mathematics with Applications 36 (3), 33 – 53.
[19] Pan, P.-Q., 2005. A revised dual projective pivot algorithm for linear programming. SIAM Journal on Optimization 16 (1), 49–68.
[20] Pan, P.-Q., 2008. A primal deficient-basis simplex algorithm for linear programming. Applied Mathematics and Computation 196, 898–912.
[21] Pan, P.-Q., Pan, Y., 2001. A phase-1 approach for the generalized simplex algorithm. Computers & Mathematics with Applications 42(1011), 1455–1464.
[22] Perold, A. F., 1980. A degeneracy exploiting LU factorization for the simplex method. Mathematical Programming 19, 239–254.
[23] Raymond, V., Soumis, F., Metrane, A., Desrosiers, J., 2010. Positive edge: A pricing criterion for the identification of non-degenerate simplex pivots, les Cahiers du GERAD G-2010-61, GERAD, Montreal, Quebec, Canada.
[24] Raymond, V., Soumis, F., Orban, D., 2010. A new version of the improved primal simplex for degenerate linear programs. Computers & Operations Research 37, 91–98.
[25] Rosat, S., Elhallaoui, I., Soumis, F., Lodi, A., 2013. Integral simplex using decomposition with primal cuts, les Cahiers du GERAD G-2013-79, GERAD, Montreal, Quebec, Canada.
[26] Towhidi, M., Desrosiers, J., Soumis, F., 2014. The positive edge criterion within COIN-OR's CLP. Computers & Operations Research 49 (0), 41 – 46.