Improved Primal Simplex: A More General Theoretical Framework and an Extended Experimental Analysis

Jérémy Omer, Samuel Rosat, Vincent Raymond, François Soumis

Submitted to *INFORMS Journal on Computing*
manuscript JOC-2014-03-OA-060

Improved Primal Simplex: A More General Theoretical Framework and an Extended Experimental Analysis

Jérémy Omer, Samuel Rosat, Vincent Raymond, François Soumis
GERAD and École Polytechnique de Montréal,
C.P. 6079, Succ. Centre-ville, Montreal (Quebec) Canada H3C 3A7
jeremy.omer@gerad.ca, samuel.rosat@gerad.ca, vincent.raymond@polymtl.ca, francois.soumis@gerad.ca

In this article, we propose a general framework for an algorithm derived from the primal simplex that guarantees a strict improvement in the objective after each iteration. Our approach relies on the identification of *compatible* variables that ensure a nondegenerate iteration if pivoted into the basis. The problem of finding a strict improvement in the objective function is proved to be equivalent to two smaller problems respectively focusing on compatible and incompatible variables. We then show that the improved primal simplex (IPS) of Elhallaoui et al. is a particular implementation of this generic theoretical framework. The resulting new description of IPS naturally emphasizes what should be considered as necessary adaptations of the framework versus specific implementation choices. This provides original insight into IPS that allows for the identification of weaknesses and potential alternative choices that would extend the efficiency of the method to a wider set of problems. We perform experimental tests on an extended collection of data sets including instances of Mittelmann's benchmark for linear programming. The results confirm the excellent potential of IPS and highlight some of its limits while showing a path toward an improved implementation of the generic algorithm.

Key words: Linear programming, simplex, degeneracy, decomposition, primal algorithms

1. Introduction

1.1. Degeneracy in the Primal Simplex

The primal simplex, described by Dantzig in 1947 (see Dantzig (1955)), was the first efficient algorithm to be developed for solving linear programs (LPs) and is still used for a large number of applications. It starts with a feasible solution obtained from a *basis* of the variable space by setting all the nonbasic variables at one of their bounds. It then

iteratively improves the objective value until optimality is reached. At each iteration, the algorithm moves to an adjacent, improving basis by selecting a nonbasic entering variable and removing a variable from the basis through a simplex *pivot*. One of the major difficulties the algorithm may encounter is called *degeneracy*. It occurs when many the basic variables are at one of their bounds. In this case, there is a high probability that the variable selected to enter the basis cannot have its value modified without making the current solution infeasible. The resulting degenerate pivot leads to no change in the current solution and no improvement in the objective value. In a study of degeneracy in the simplex method, Perold (1980) states that typically if on average 20% of the basic variables are at one of their bounds, 50% of the iterations will be degenerate. Combined with the multiplicity of highly degenerate problems arising in industrial applications, this observation highlights the need for an efficient treatment of degeneracy in any good implementation of the primal simplex algorithm.

1.2. Dealing with Primal Degeneracy: A Short State of the Art

Several techniques have been developed to cope with degeneracy. One family of these methods focuses on the selection of the variable entering the basis. For instance, in Greenberg (1978), a vector is computed such that each of its positive entries corresponds to a nonbasic variable that will necessarily lead to a degenerate pivot if it is chosen to enter the basis. The number of operations is similar to the computation of a reduced cost, but the test is only heuristic since it does not identify all the nonbasic variables that would lead to a degenerate pivot. The perturbation (Benichou et al. 1977) and bound-shifting (Gill et al. 1989) techniques follow a different path. They apply a random modification of either the values of the degenerate variables or their bounds to put an end to a sequence of nonimproving pivots. The random modification avoids performing iterations without improvement, but it usually replaces them with small steps. Although not supported by a strong theory, the two methods, that mostly differ from an implementation point of view, perform well, and they appear in most efficient simplex codes. For further information, an excellent study of degeneracy emphasizing its computational aspects is given by Maros (2003).

Another approach aims to take advantage of degeneracy rather than just to minimize its negative effects. Geometrically, a degenerate vertex of the n-dimensional polytope described by the LP's constraints is the crossing point of more than n facets of the polytope. Degeneracy thus corresponds to a local excess of information, which suggests that

a smaller problem may be considered locally to make progress from a degenerate solution. Perold (1980) introduces a particular degeneracy structure in the LU decomposition of the basis, which involves fewer calculations when performing degenerate pivots. Pan (2008) generalizes the definition of basis to include *deficient bases* containing p independent columns, with p lower than the number of rows. When degeneracy occurs, a deficient basis, smaller than the usual square basis matrix, may be used to perform the pivot calculations. Generalizing the dynamic constraint aggregation described by Elhallaoui et al. (2005) for set partitioning problems, the improved primal simplex (IPS) of Elhallaoui et al. (2011) makes the most of degeneracy by simultaneously reducing the number of rows and columns to perform as many cheap pivots as possible. Once all the interesting pivots are done, a *complementary* LP is solved to find a dual solution maximizing the minimum reduced cost. Either this reduced cost is null and optimality is proved, or it corresponds to a combination of primal variables that may enter the basis through a sequence of pivots ending with a strict improvement in the objective. Metrane et al. (2010) prove the equivalence of IPS to a particular column generation algorithm by respectively identifying the reduced and complementary problems of IPS with the master and pricing problems of column generation.

1.3. Contribution Statement

The present article is motivated by the promising experimental results for IPS that are reported by Elhallaoui et al. (2011) and Raymond et al. (2010b). However, Elhallaoui et al. (2011) conduct their tests on instances containing a majority of set-partitioning constraints, while the improvements proposed in Raymond et al. (2010b) are designed for only a part of this restricted benchmark. Based on this proof of concept, our intent is to investigate the performance of the algorithm on a much more diversified benchmark and, using a new, more general theoretical description of the algorithm, to highlight potential improvements.

With that in mind, our first contribution is to describe IPS for an LP with upper and lower bounds on the variables. In Elhallaoui et al. (2011), the algorithm is described for an LP in standard form, because every LP has an equivalent standard form, but this is not compatible with an efficient implementation. Including the bounds explicitly shows how degeneracy can be efficiently handled when a variable may be at either its lower or its upper bound, and it specifies the special treatment that unbounded variables should

receive. The presentation of the algorithm is then generalized to a theoretical framework of which IPS is only one possible implementation. Our goal is to emphasize the steps in IPS that may be seen as implementation choices. By analyzing these choices independently of the core of the method, we can identify their main advantages and drawbacks and consider alternative implementations. Moreover, the new framework is purely primal, which simplifies the presentation and the proofs. We then extend the benchmark by including larger instances and problems that do not share the specific structure of the instances used by Elhallaoui et al. (2011) and Raymond et al. (2010b). A thorough analysis of the results provides a much more accurate description of the families of problems that may be solved efficiently using IPS while highlighting the implementation choices that would enlarge its domain of efficiency. The main contribution is thus to identify the limits of IPS while laying the theoretical foundation for future improvements.

The generic theoretical framework is developed in Section 2, and the implementation choices for IPS are described in Section 3. The results of the experimental tests are analyzed in Section 4, and in Section 5 we discuss directions for future research.

1.4. Notation

Lower-case bold symbols are used for column vectors and upper-case bold symbols denote matrices. For subsets $\mathcal{I} \subseteq \{1, ..., m\}$ of row indices and $\mathcal{J} \subseteq \{1, ..., n\}$ of column indices, the submatrix of $\boldsymbol{A}$ with rows indexed by $\mathcal{I}$ and columns indexed by $\mathcal{J}$ is denoted $\boldsymbol{A}_{\mathcal{I}\mathcal{J}}$. Similarly, $\boldsymbol{A}_{\mathcal{I}\cdot}$ is the set of rows of $\boldsymbol{A}$ indexed by $\mathcal{I}$, $\boldsymbol{A}_{\cdot\mathcal{J}}$ is the set of columns of $\boldsymbol{A}$ indexed by $\mathcal{J}$, and for any vector $\boldsymbol{v} \in \mathbb{R}^n$, $\boldsymbol{v}_{\mathcal{J}}$ is the subvector of all v_j, $j \in \mathcal{J}$. The vector of all zeros with dimension dictated by the context is denoted $\mathbf{0}$, and $\boldsymbol{A}^T$ is the transpose of $\boldsymbol{A}$.

2. A Generic Decomposition Algorithm Taking Advantage of Degeneracy

Like the primal simplex or gradient-descent methods, IPS is a primal algorithm: it starts from an initial feasible solution and iteratively improves it until optimality is reached. Primal algorithms mainly differ from each other in the improvement procedure, i.e., the way of finding a better solution. Some of them guarantee a strict improvement in the objective function, whereas others do not. Typically, Dantzig's primal simplex cannot guarantee this strict improvement when the instances are degenerate. As for IPS, it was designed to avoid nonimproving steps, although in practice degenerate pivots are still performed. The

purpose of this section is to describe a general decomposition algorithm yielding a strict improvement at each step; IPS may be seen as one of its possible implementations. Moreover, we give a theoretical insight based on a purely primal approach. This approach both gives a rather geometric understanding of the algorithm and eases the proof of optimality. In Section 2.1, we describe a primal algorithm that guarantees a strict improvement at each iteration, and in Section 2.2, we show how the algorithm can be improved by decomposing the problem.

2.1. A Primal Algorithm for Degenerate Problems

IPS was designed to start from an extreme solution of the feasible domain, assuming nonnegative variables. We generalize it to allow any feasible starting solution, and we consider an LP with bounded variables:

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

where $\boldsymbol{x} \in \mathbb{R}^n$ is the vector of decision variables, $\boldsymbol{c} \in \mathbb{R}^n$ is the cost vector, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ is the constraint matrix, $\boldsymbol{b} \in \mathbb{R}^m$ is the right-hand side vector, and $\boldsymbol{l}, \boldsymbol{u} \in \mathbb{R} \cup \{\pm\infty\}$, $\boldsymbol{l} < \boldsymbol{u}$, are respectively the lower- and upper-bound vectors. We assume that $\boldsymbol{A}$ is of full rank m with $m \le n$ and has no zero-column, and that the feasible domain $\mathcal{F}_\mathrm{P}$ is nonempty.

Given a feasible solution $\boldsymbol{x} \in \mathcal{F}_\mathrm{P}$, the indices of its components can be partitioned into four sets, $\mathcal{P}$, $\mathcal{L}$, $\mathcal{U}$, and $\mathcal{M}$. $\mathcal{L}$ and $\mathcal{U}$ respectively designate the set of indices of variables that are at their lower and upper bounds. For the remaining variables—all of them being strictly within their bounds—$\mathcal{P}$ designates a maximal subset such that $\boldsymbol{A}_{\cdot\mathcal{P}}$ is linearly independent, while $\mathcal{M}$ gathers all the others. Note that $|\mathcal{P}| \le m$, but nothing can be inferred for $|\mathcal{M}|$. With this decomposition and a convenient reordering of the columns, $\boldsymbol{x} = (\boldsymbol{x}_\mathcal{P}, \boldsymbol{x}_\mathcal{L}, \boldsymbol{x}_\mathcal{U}, \boldsymbol{x}_\mathcal{M})$ with $\boldsymbol{l}_\mathcal{P} < \boldsymbol{x}_\mathcal{P} < \boldsymbol{u}_\mathcal{P}$, $\boldsymbol{x}_\mathcal{L} = \boldsymbol{l}_\mathcal{L}$, $\boldsymbol{x}_\mathcal{U} = \boldsymbol{u}_\mathcal{U}$, and $\boldsymbol{l}_\mathcal{M} < \boldsymbol{x}_\mathcal{M} < \boldsymbol{u}_\mathcal{M}$, and

$$\boldsymbol{A}_{\cdot\mathcal{P}}\boldsymbol{x}_\mathcal{P} + \boldsymbol{A}_{\cdot\mathcal{L}}\boldsymbol{l}_\mathcal{L} + \boldsymbol{A}_{\cdot\mathcal{U}}\boldsymbol{u}_\mathcal{U} + \boldsymbol{A}_{\cdot\mathcal{M}}\boldsymbol{x}_\mathcal{M} = b. \tag{1}$$

$\boldsymbol{A}_{\cdot\mathcal{P}}$ is a basis of the linear span of $\boldsymbol{A}_{\cdot\mathcal{P}\cup\mathcal{M}}$, denoted $\mathrm{Span}(\boldsymbol{A}_{\cdot\mathcal{P}\cup\mathcal{M}})$, and $\mathcal{P}$ is therefore called the *reduced* or *working basis*. Its indices and the variables of $\boldsymbol{x}_\mathcal{P}$ are respectively referred to as *basic indices* and *basic variables*, and $p = |\mathcal{P}|$ denotes the cardinality of this working basis. Finally, the set of all nonbasic indices is denoted $\mathcal{N}$: $\mathcal{N} = \mathcal{L} \cup \mathcal{U} \cup \mathcal{M}$.

In the rest of this paper, $\boldsymbol{x}^0 \in \mathcal{F}_\mathrm{P}$ denotes the current feasible solution of (P).

DEFINITION 1. $\boldsymbol{d} \in \mathbb{R}^n$ is a *feasible* direction at $\boldsymbol{x}^0$ if there exists a step $\rho > 0$ such that $\boldsymbol{x}^0 + \rho \cdot \boldsymbol{d} \in \mathcal{F}_{\mathrm{P}}$. Moreover, $\boldsymbol{d}$ is *improving* if $\boldsymbol{c}^T\boldsymbol{d} < 0$, i.e., if taking a positive step along $\boldsymbol{d}$ yields an improvement in the objective value.

To guarantee a strict improvement, the following problem must be addressed:

Improvement Problem (IMP): Given a feasible solution $\boldsymbol{x}^0$, either supply a feasible improving direction $\boldsymbol{d} \in \mathbb{R}^n$, or assert that $\boldsymbol{x}^0$ is optimal for (P).

The feasible improving direction $\boldsymbol{d}$ can be normalized at will as long as the length of the step ρ is chosen such that $\boldsymbol{x}^0 + \rho \cdot \boldsymbol{d} \in \mathcal{F}_{\mathrm{P}}$. The set of all feasible directions at $\boldsymbol{x}^0$ can then be described as the following cone section:

$$\Delta^0 = \left\{ \boldsymbol{d} \in \mathbb{R}^n \,|\, \boldsymbol{A}\boldsymbol{d} = 0\,,\, \boldsymbol{d}_{\mathcal{U}} \leq 0\,,\, \boldsymbol{d}_{\mathcal{L}} \geq 0\,,\, \sum_{i \in \mathcal{P} \cup \mathcal{N}} w_i \,|d_i| = 1 \right\} \tag{2}$$

where $\boldsymbol{w} \in \mathbb{R}^n$, $\boldsymbol{w}_{\mathcal{P}} \geq 0$, $\boldsymbol{w}_{\mathcal{N}} > 0$. $\sum_{i \in \mathcal{P} \cup \mathcal{N}} \boldsymbol{w}_i |\boldsymbol{d}_i| = 1$ geometrically represents a normalization constraint. Without this constraint, Δ^0 is a cone and if $\boldsymbol{x}^0$ is an extreme point of $\mathcal{F}_{\mathrm{P}}$, the extreme directions of this cone are the directions of the edges of $\mathcal{F}_{\mathrm{P}}$ at $\boldsymbol{x}^0$.

Since $\boldsymbol{d}_{\mathcal{U}} \leq 0$ and $\boldsymbol{d}_{\mathcal{L}} \geq 0$, the normalization constraint can be written in a way that makes it closer to a linear equality, which will prove useful later. With $\boldsymbol{w}_{\mathcal{U}} < 0$, $\boldsymbol{w}_{\mathcal{L}} > 0$, $\boldsymbol{w}_{\mathcal{M}} > 0$ and $\boldsymbol{w}_{\mathcal{P}} \geq 0$, Equation (2) reads:

$$\Delta^0 = \left\{ \boldsymbol{d} \in \mathbb{R}^n \,|\, \boldsymbol{A}\boldsymbol{d} = 0\,,\, \boldsymbol{d}_{\mathcal{U}} \leq 0\,,\, \boldsymbol{d}_{\mathcal{L}} \geq 0\,,\, \sum_{i \in \mathcal{P} \cup \mathcal{M}} w_i \,|d_i| + \sum_{i \in \mathcal{U} \cup \mathcal{L}} w_i \,|d_i| = 1 \right\}. \tag{3}$$

Moreover, given a feasible improving direction $\boldsymbol{d}^0$, the *maximal step* along $\boldsymbol{d}^0$ at $\boldsymbol{x}^0$, defined as $\rho^0 = \max\left\{\rho | \boldsymbol{x}^0 + \rho \boldsymbol{d}^0 \in \mathcal{F}_{\mathrm{P}}\right\}$, is the most interesting step to take since it yields the greatest possible improvement along $\boldsymbol{d}^0$.

One way to tackle the improvement problem is to minimize the chosen objective over the domain Δ^0. Typically, that objective is related to the improvement in the objective function given by $\boldsymbol{d} \in \Delta^0$. IMP may thus come down to solving the following *greatest normalized improvement* program:

$$z^\star_{\mathrm{GNI}} = \min_{\boldsymbol{d} \in \mathbb{R}^n} \left\{ \boldsymbol{c}^T \boldsymbol{d} \,|\, \boldsymbol{d} \in \Delta^0 \right\}. \tag{GNI}$$

LEMMA 1. *x^0 is optimal for* (P) *if and only if* $z^\star_{\mathrm{GNI}} \geq 0$.

Based on Lemma 1, the procedure of Algorithm 1 is guaranteed to reach optimality. Moreover, each step $\rho^k \cdot \boldsymbol{d}^k$ taken during this procedure yields a strict improvement in the objective function since all the improving directions $\boldsymbol{d}^k$ are feasible, thus $\rho^k > 0$ and each improvement is $z^{k+1} - z^k = \rho^k \cdot \boldsymbol{c}^T \boldsymbol{d}^k < 0$. Degeneracy is therefore avoided.

Algorithm 1: Nondecomposed Primal Improvement Procedure

Input: A linear program P, $\boldsymbol{x}^0$ a feasible solution of P.

Output: $\boldsymbol{x}^k$, an optimal solution of P.

$k \leftarrow 0$; $z^\star_{\text{GNI}} \leftarrow -1$;

while $z^\star_{\text{GNI}} < 0$ **do**

  Solve GNI; $z^\star_{\text{GNI}} \leftarrow$ optimal objective value of GNI;

  $\boldsymbol{d}^k \leftarrow$ optimal solution of GNI; $\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{x}^k + \rho^k \cdot \boldsymbol{d}^k$; $k \leftarrow k+1$;

2.2. Of Compatibility and Decomposition: Turning Degeneracy into an Asset

Algorithm 1 not only avoids degeneracy but it represents an opportunity to turn it into an asset. GNI may be decomposed into two smaller problems that, even if they are solved separately, assert that the current solution $\boldsymbol{x}^k$ is optimal or provide a feasible improving direction. This decomposition is the main topic of the present subsection. The main idea is to sequentially find nondegenerate improving simplex pivots if they exist, and otherwise to solve a more complicated problem similar to GNI. This decomposition relies on the notion of *compatibility*.

DEFINITION 2. Given a working basis $\mathcal{P}$, a vector $\boldsymbol{v} \in \mathbb{R}^m$ is *compatible* (with $\mathcal{P}$) if and only if $\boldsymbol{v} \in \text{Span}(\boldsymbol{A}_{\cdot\mathcal{P}})$.

This notion is extended to the nonbasic columns of $\boldsymbol{A}_{\cdot\mathcal{N}}$, their indices, and the corresponding variables. The set of indices of the *compatible* columns of $\boldsymbol{A}_{\cdot\mathcal{N}}$ is denoted $\mathcal{C}$. Every other column of $\mathcal{I} = \mathcal{N} \setminus \mathcal{C}$ is said to be *incompatible*, and $(\mathcal{C}, \mathcal{I})$ forms a partition of $\mathcal{N}$.

PROPOSITION 1. $\mathcal{M} \subset \mathcal{C}$.

Proof By definition of $\mathcal{P}$, for every index $j \in \mathcal{M}$, the corresponding column $\boldsymbol{A}_{\cdot j}$ is in $\text{Span}(\boldsymbol{A}_{\cdot\mathcal{P}})$. □

According to Proposition 1, $\mathcal{C} = \mathcal{C}_\mathcal{L} \cup \mathcal{C}_\mathcal{U} \cup \mathcal{M}$ and $\mathcal{I} = \mathcal{I}_\mathcal{L} \cup \mathcal{I}_\mathcal{U}$, where for any pair of sets $(\mathcal{X}, \mathcal{Y})$, $\mathcal{X}_\mathcal{Y} = \mathcal{X} \cap \mathcal{Y}$. All incompatible variables are thus at their lower or upper bounds. The following proposition indicates the value of compatible columns when solving IMP.

PROPOSITION 2. *Compatible columns are exactly those that yield nondegenerate pivots if inserted into the working basis* $\mathcal{P}$.

Proof Let $j \in \mathcal{N}$. Pivoting $\boldsymbol{A}_{\cdot j}$ into the working basis leads to a solution $\boldsymbol{x}^1 \in \mathcal{F}_\text{P}$ (not necessarily different from $\boldsymbol{x}^0$). Since only $\boldsymbol{A}_{\cdot j}$ has been pivoted, for all $k \in \mathcal{N} \setminus \{j\}$,

$\boldsymbol{x}^1_k = \boldsymbol{x}^0_k$. Since $\boldsymbol{x}^1$ and $\boldsymbol{x}^0$ both satisfy the linear constraints of (P), $\boldsymbol{Ax} = \boldsymbol{b}$, and thus $\boldsymbol{A}_{.\mathcal{P}}\boldsymbol{x}^0_{\mathcal{P}} + \boldsymbol{A}_{.j}x^0_j = \boldsymbol{A}_{.\mathcal{P}}\boldsymbol{x}^1_{\mathcal{P}} + \boldsymbol{A}_{.j}x^1_j$, so if $\boldsymbol{d} = \boldsymbol{x}^1 - \boldsymbol{x}^0$

$$\boldsymbol{A}_{.j}d_j = -\boldsymbol{A}_{.\mathcal{P}}\boldsymbol{d}_{\mathcal{P}}. \tag{4}$$

Since $\boldsymbol{A}_{.\mathcal{P}}$ is of full rank, $\boldsymbol{d}_{\mathcal{P}} = \boldsymbol{0}$ if and only if $d_j = 0$. Therefore, $\boldsymbol{A}_{.j}$ yields a nondegenerate pivot if and only if $\boldsymbol{x}^1 \neq \boldsymbol{x}^0$, that is, if and only if $d_j \neq 0$. Finally, Equation (4) admits a solution $\boldsymbol{d}$ such that $d_j \neq 0$ if and only if $\boldsymbol{A}_{.j} \in \mathrm{Span}(\boldsymbol{A}_{.\mathcal{P}})$, so the result holds. □

Note that bounds are not mentioned in the above proof. Since all the variables in $\mathcal{P}$ are strictly within their bounds, they can increase or decrease without violating their bounds. Therefore, if $|d_j|$ is sufficiently small, for all $i \in \mathcal{P}$, $l_i \leq x^1_i \leq u_i$.

From the perspective of the simplex algorithm, it will be efficient to give priority to the compatible columns since they are exactly those that yield nondegenerate pivots when inserted into the working basis $\mathcal{P}$. Based on the compatible-incompatible partition of the variables, we will show in the following that GNI can be decomposed into the *Reduced*-GNI problem (R-GNI) and the *Complementary*-GNI problem (C-GNI) that respectively focus on the compatible and incompatible variables:

$$\begin{aligned} z^\star_{\text{R-GNI}} = \min_{(\boldsymbol{d}_{\mathcal{P}}, \boldsymbol{d}_{\mathcal{C}})} \; & \boldsymbol{c}^T_{\mathcal{P}}\boldsymbol{d}_{\mathcal{P}} && + \boldsymbol{c}^T_{\mathcal{C}}\boldsymbol{d}_{\mathcal{C}} \\ \text{s.t.} \; & \boldsymbol{A}_{.\mathcal{P}}\boldsymbol{d}_{\mathcal{P}} && + \boldsymbol{A}_{.\mathcal{C}}\boldsymbol{d}_{\mathcal{C}} && = 0 \\ & \textstyle\sum_{i\in\mathcal{P}} w_i\,|d_i| + \sum_{i\in\mathcal{M}} w_i\,|d_i| + \sum_{i\in\mathcal{C}_{\mathcal{U}}\cup\mathcal{C}_{\mathcal{L}}} w_i d_i && && = 1 \\ & \boldsymbol{d}_{\mathcal{C}_{\mathcal{L}}} \geq 0 && , \; \boldsymbol{d}_{\mathcal{C}_{\mathcal{U}}} \leq 0 \end{aligned} \tag{R-GNI}$$

$$\begin{aligned} z^\star_{\text{C-GNI}} = \min_{(\boldsymbol{d}_{\mathcal{P}}, \boldsymbol{d}_{\mathcal{I}})} \; & \boldsymbol{c}^T_{\mathcal{P}}\boldsymbol{d}_{\mathcal{P}} && + \boldsymbol{c}^T_{\mathcal{I}}\boldsymbol{d}_{\mathcal{I}} \\ \text{s.t.} \; & \boldsymbol{A}_{.\mathcal{P}}\boldsymbol{d}_{\mathcal{P}} && + \boldsymbol{A}_{.\mathcal{I}}\boldsymbol{d}_{\mathcal{I}} && = 0 \\ & \textstyle\sum_{i\in\mathcal{P}} w_i\,|d_i| + \sum_{i\in\mathcal{M}} w_i d_i && && = 1 \\ & \boldsymbol{d}_{\mathcal{I}_{\mathcal{L}}} \geq 0 && , \; \boldsymbol{d}_{\mathcal{I}_{\mathcal{U}}} \leq 0 \end{aligned} \tag{C-GNI}$$

R-GNI and C-GNI are both smaller than GNI, so solving them separately must be much faster than solving GNI. The only remaining issue is to ensure that the search space for an improving direction Δ^0 can be split into that of the compatible columns ($\mathcal{F}_{\text{R-GNI}}$) and that of the incompatible columns ($\mathcal{F}_{\text{C-GNI}}$), i.e., that solving GNI is equivalent to solving R-GNI and C-GNI.

LEMMA 2. *A column* $\boldsymbol{A}_{\cdot j}$ *is compatible if and only if there exists* $\boldsymbol{v} \in \mathbb{R}^p$ *such that* $\boldsymbol{A}_{\cdot j} = \boldsymbol{A}_{\cdot \mathcal{P}}\boldsymbol{v}$. *In this case,* $\boldsymbol{v}$ *is unique.*

Proof This lemma is a consequence of the definition of compatibility and of the fact that $\boldsymbol{A}_{\cdot \mathcal{P}}$ is a full-rank matrix. □

THEOREM 1. *Assume* $\boldsymbol{w}_{\mathcal{P}} = 0$, *then* $z^\star_{\text{GNI}} = \min\{z^\star_{\text{R-GNI}}, z^\star_{\text{C-GNI}}\}$.

Proof Recall that $\boldsymbol{w}_{\mathcal{L}} > 0$, $\boldsymbol{w}_{\mathcal{U}} < 0$, $\mathcal{C} = \mathcal{C}_{\mathcal{L}} \cup \mathcal{C}_{\mathcal{U}} \cup \mathcal{M}$, and $\mathcal{I} = \mathcal{I}_{\mathcal{L}} \cup \mathcal{I}_{\mathcal{U}}$. Let $\boldsymbol{d} = (\boldsymbol{d}_{\mathcal{P}}, \boldsymbol{d}_{\mathcal{C}}, \boldsymbol{d}_{\mathcal{I}})$ be a solution of GNI. If $\boldsymbol{d}_{\mathcal{C}} = \boldsymbol{0}$ or $\boldsymbol{d}_{\mathcal{I}} = \boldsymbol{0}$, $\boldsymbol{d}$ is respectively a solution of C-GNI or R-GNI and the result clearly holds.

Suppose now that $\boldsymbol{d}_{\mathcal{C}} \neq \boldsymbol{0}$ and $\boldsymbol{d}_{\mathcal{I}} \neq \boldsymbol{0}$. We first prove that $\boldsymbol{d}$ can be written as a convex combination of $\boldsymbol{u}'$, the solution of R-GNI, and $\boldsymbol{v}'$, the solution of C-GNI. For every $j \in \mathcal{C}$, $\boldsymbol{A}_{\cdot j} \in \text{Span}(\boldsymbol{A}_{\cdot P})$, so linear combinations of compatible columns $\boldsymbol{A}_{\cdot \mathcal{C}}\boldsymbol{d}_{\mathcal{C}}$ are also compatible. According to Lemma 2, there exists $\boldsymbol{u}_{\mathcal{P}} \in \mathbb{R}^{|\mathcal{P}|}$ such that $\boldsymbol{A}_{\cdot \mathcal{P}}\boldsymbol{u}_{\mathcal{P}} = -\boldsymbol{A}_{\cdot \mathcal{C}}\boldsymbol{d}_{\mathcal{C}}$. Let $\boldsymbol{u} = (\boldsymbol{u}_{\mathcal{P}}, \boldsymbol{d}_{\mathcal{C}}, \boldsymbol{0})$ and $\boldsymbol{v} = \boldsymbol{d} - \boldsymbol{u} = (\boldsymbol{v}_{\mathcal{C}}, \boldsymbol{0}, \boldsymbol{d}_{\mathcal{I}})$. Let $\alpha_{\boldsymbol{u}} = \sum_{i \in \mathcal{M}} w_i |d_i| + \sum_{i \in \mathcal{C}_{\mathcal{U}} \cup \mathcal{C}_{\mathcal{L}}} w_i d_i$ and $\alpha_{\boldsymbol{v}} = \boldsymbol{w}^T\boldsymbol{v} = 1 - \alpha_{\boldsymbol{u}}$. Thus, $0 < \alpha_{\boldsymbol{u}}, \alpha_{\boldsymbol{v}} < 1$. Moreover, let $\boldsymbol{u}' = \boldsymbol{u}/\alpha_{\boldsymbol{u}}$ and $\boldsymbol{v}' = \boldsymbol{v}/\alpha_{\boldsymbol{v}}$ be the corresponding normalized directions. With these definitions, $\boldsymbol{d} = \alpha_{\boldsymbol{u}}\boldsymbol{u}' + \alpha_{\boldsymbol{v}}\boldsymbol{v}'$. By construction, $\boldsymbol{u}'$ (resp. $\boldsymbol{v}'$) is a solution of R-GNI (resp. C-GNI) and $\alpha_{\boldsymbol{u}} + \alpha_{\boldsymbol{v}} = 1$, because $\boldsymbol{d} \in \Delta^0$. Therefore, $\boldsymbol{d} = \alpha_{\boldsymbol{u}}\boldsymbol{u}' + \alpha_{\boldsymbol{v}}\boldsymbol{v}'$ is a convex combination of a solution of R-GNI ($\boldsymbol{u}'$) and a solution of C-GNI ($\boldsymbol{v}'$).

Looking at the objective function, the convex combination reads $\boldsymbol{c}^T\boldsymbol{d} = \alpha_{\boldsymbol{u}}(\boldsymbol{c}^T\boldsymbol{u}') + \alpha_{\boldsymbol{v}}(\boldsymbol{c}^T\boldsymbol{v}')$. Either $\boldsymbol{c}^T\boldsymbol{d} \geq \boldsymbol{c}^T\boldsymbol{u}'$ or $\boldsymbol{c}^T\boldsymbol{d} \geq \boldsymbol{c}^T\boldsymbol{v}'$. However, since every solution of R-GNI and C-GNI is also a solution of GNI and $\boldsymbol{d}$ is optimal for GNI, $\boldsymbol{c}^T\boldsymbol{d} \leq \boldsymbol{c}^T\boldsymbol{u}'$ and $\boldsymbol{c}^T\boldsymbol{d} \leq \boldsymbol{c}^T\boldsymbol{v}'$. One of the two inequalities is thus an equality and the second follows from $\boldsymbol{c}^T\boldsymbol{d} = \alpha_{\boldsymbol{u}}(\boldsymbol{c}^T\boldsymbol{u}') + \alpha_{\boldsymbol{v}}(\boldsymbol{c}^T\boldsymbol{v}')$. Therefore, $\boldsymbol{c}^T\boldsymbol{d} = \boldsymbol{c}^T\boldsymbol{u}' = \boldsymbol{c}^T\boldsymbol{v}'$, and since $\boldsymbol{d}$ is an optimal solution of GNI, $z^\star_{\text{GNI}} = z^\star_{\text{R-GNI}} = z^\star_{\text{C-GNI}}$. □

Theorem 1 extends a theorem established by Rosat et al. (2014) for the set partitioning problem to general linear programming, and it has no equivalent in past presentations of IPS. It gives strong theoretical support to the compatible-incompatible partition and to the R-GNI/C-GNI decomposition of GNI: it states that one of the optimal solutions of GNI is an optimal solution of one of the decomposed problems. Solving them separately is thus equivalent to solving GNI, i.e., to determining whether or not $\boldsymbol{x}^0$ is optimal (see Lemma 1). Since the theorem requires that $\boldsymbol{w}_{\mathcal{P}} = 0$, we retain this assumption in the rest of

this paper. This last condition is not overly restrictive. It is only natural that the emphasis is put on the nonbasic variables during the search for an improving direction. The same assumption is made by Elhallaoui et al. (2011) and by most pricing rules designed for the simplex algorithm.

COROLLARY 1. *If, for a current solution* $\boldsymbol{x}$ *and the associated partition* $(\mathcal{P},\mathcal{C},\mathcal{I})$*, the optimal values of* R-GNI *and* C-GNI *are both nonnegative, then* $\boldsymbol{x}$ *is optimal for* (P).

This corollary is a straightforward consequence of Theorem 1 and Lemma 1 .

One other key aspect of the decomposition is that R-GNI may be further reduced by observing that the rank of its constraint matrix $\boldsymbol{A}_{\cdot\mathcal{P}\cup\mathcal{C}}$ is the same as the rank of $\boldsymbol{A}_{\cdot\mathcal{P}}$, i.e., p. Since R-GNI is feasible by construction, it contains $m-p$ redundant constraints that may be removed without harm. Assuming that the rows are permuted so that the first p constraints are independent, these rows are indexed by $\mathcal{P}$ while the redundant ones have their index set denoted $\overline{\mathcal{P}} = \{p+1,\ldots,m\}$. Since we also have $\boldsymbol{w}_{\mathcal{P}} = 0$, R-GNI may be equivalently rewritten:

$$
\begin{aligned}
z^{\star}_{\text{R-GNI}} = \min_{(\boldsymbol{d}_{\mathcal{P}},\boldsymbol{d}_{\mathcal{C}})} \ & \boldsymbol{c}_{\mathcal{P}}^T\boldsymbol{d}_{\mathcal{P}} \quad + \boldsymbol{c}_{\mathcal{C}}^T\boldsymbol{d}_{\mathcal{C}} \\
\text{s.t.} \ & \boldsymbol{A}_{\mathcal{P}\mathcal{P}}\boldsymbol{d}_{\mathcal{P}} + \boldsymbol{A}_{\mathcal{P}\mathcal{C}}\boldsymbol{d}_{\mathcal{C}} = 0 \\
& \textstyle\sum_{i\in\mathcal{M}} w_i\,|d_i| + \sum_{i\in\mathcal{C}_{\mathcal{U}}\cup\mathcal{C}_{\mathcal{L}}} w_i d_i = 1 \\
& \boldsymbol{d}_{\mathcal{C}_{\mathcal{L}}} \geq 0 \ , \ \boldsymbol{d}_{\mathcal{C}_{\mathcal{U}}} \leq 0
\end{aligned}
$$

This new form clearly emphasizes the potential advantage of solving R-GNI instead of GNI to find an improving feasible direction.

The previous results canonically lead to the procedure described in Algorithm 2. The main idea is to first look at compatible columns to perform nondegenerate pivots (R-GNI), and if no suitable columns are found, to look for a collection of incompatible columns that globally supplies an improvement (C-GNI).

Corollary 1 shows that $\boldsymbol{x}$ is optimal if and only if $z^{\star}_{\text{R-GNI}} \geq 0$ and $z^{\star}_{\text{C-GNI}} \geq 0$, therefore the main while loop ends when this condition is satisfied. Notice however that this criterion is valid only if R-GNI and C-GNI refer to the same partition $(\mathcal{P},\mathcal{C},\mathcal{I})$. $z^{\star}_{\text{R-GNI}}$ and $z^{\star}_{\text{C-GNI}}$ are then set to an arbitrary negative value after the updates of $(\mathcal{P},\mathcal{C},\mathcal{I})$ (lines 7 and 12) to exit the main loop only when both values have been computed for the same partition.

Algorithm 2: Compatibility-based Decomposed Primal Improvement Procedure

Input: A linear program P, $\boldsymbol{x}^0$ a feasible solution of P.

Output: $\boldsymbol{x}$, an optimal solution of P.

1 $\boldsymbol{x} \leftarrow \boldsymbol{x}^0$; $(\mathcal{P},\mathcal{C},\mathcal{I}) \leftarrow$ partition corresponding to $\boldsymbol{x}$; $z^\star_{\text{R-GNI}} \leftarrow -1$; $z^\star_{\text{C-GNI}} \leftarrow -1$;
2 **while** $z^\star_{\text{R-GNI}} < 0$ *or* $z^\star_{\text{C-GNI}} < 0$ **do**
3 **while** *some criterion is satisfied and* $z^\star_{\text{R-GNI}} < 0$ **do**
4 Solve R-GNI; $z^\star_{\text{R-GNI}} \leftarrow$ the optimal objective value of R-GNI;
5 **if** $z^\star_{\text{R-GNI}} < 0$ **then**
6 Update $\boldsymbol{x}$ by following the direction found by R-GNI; update $(\mathcal{P},\mathcal{C},\mathcal{I})$;
7 $z^\star_{\text{C-GNI}} \leftarrow -1$;
8 **while** *some criterion is satisfied and* $z^\star_{\text{C-GNI}} < 0$ **do**
9 Solve C-GNI; $z^\star_{\text{C-GNI}} \leftarrow$ the optimal objective value of C-GNI;
10 **if** $z^\star_{\text{C-GNI}} < 0$ **then**
11 Update $\boldsymbol{x}$ by following the direction found by C-GNI; update $(\mathcal{P},\mathcal{C},\mathcal{I})$;
12 $z^\star_{\text{R-GNI}} \leftarrow -1$;

REMARK 1. No theoretical ground requires R-GNI to be given priority over C-GNI. The order in which both while loops 3-7 and 8-12 are performed in Algorithm 2 is based on common sense: R-GNI is easier to solve than C-GNI because compatible columns guarantee nondegenerate pivots. These two while-loops could possibly be switched. Also, the criteria that appear at lines 3 and 8 may be set to allow for several instances of R-GNI or C-GNI to be solved consecutively. They may thus be used to emphasize the preference for one or the other problem.

Algorithm 2 provides a strong basis for an efficient algorithm for degenerate LPs. Yet it is too generic to be implemented as such, and choices are necessary to make it a practical LP solver. Matters such as how to find an initial solution, when to update the $(\mathcal{P},\mathcal{C},\mathcal{I})$ partition, or when to switch from solving R-GNI to solving C-GNI are more than just implementation details. They are critical features of the algorithm that can dramatically change its computational behavior and the resulting running time. The next section thoroughly addresses the evolution from this theoretical scheme to a numerically efficient procedure.

3. Implementing the Framework: IPS

Implementations similar to Algorithm 2 already exist, even though the associated generic theory was never presented. One of them is, as already mentioned, the IPS algorithm

of Elhallaoui et al. (2011). IPS includes practical implementation techniques, but it also makes implicit choices. In particular, all the weights in the normalization constraint are set to 1, and the complementary problem is transformed through a costly algebraic operation to reduce its size. These choices are discussed below, together with the most important implementation issues. Although some of them are discussed in Elhallaoui et al. (2011) and Raymond et al. (2010b), we adapt the presentation to emphasize the relationship between IPS and the aforementioned generic framework. New details are also needed because of the efficient treatment of upper-bounded and unbounded variables. Moreover, our intent to diversify the benchmark and the general view provided by the previous section give rise to several improvements.

3.1. Starting the Algorithm with an Initial Feasible Solution

One important improvement is that the theoretical presentation in the previous section does not assume that the initial feasible solution is basic. We use a simplex phase I to generate the initial solution in our tests, but IPS could start from a feasible solution generated, for instance, by an external heuristic procedure. The only additional work is to identify a maximal linearly independent set of columns within their bounds. This can be done, for instance, by performing Gaussian eliminations until reduction to row echelon form. The operation simultaneously provides the set $\mathcal{P}$ and the $m-p$ redundant constraints of R-Gni.

3.2. Identifying Compatible Variables and Transforming the Complementary Problem

The efficiency of Algorithm 2 relies on the ability to quickly update the partition $(\mathcal{P},\mathcal{C},\mathcal{I})$. Once $\mathcal{P}$ is known, identifying $\mathcal{C}$ is equivalent to finding the set of nonbasic columns that belong to $\text{Span}(\boldsymbol{A}_{\cdot\mathcal{P}})$. A straightforward way to do this is to complete the columns of $\boldsymbol{A}_{\cdot\mathcal{P}}$ with vectors of $\mathbb{R}^m$ to form a basis matrix $\boldsymbol{B}$ of $\mathbb{R}^m$. Since the columns of $\boldsymbol{A}_{\cdot\mathcal{P}}$ are the first p columns, any vector $\boldsymbol{v}\in\mathbb{R}^m$ is written in the basis matrix $\boldsymbol{B}$ as $\left(\left(\boldsymbol{B}^{-1}\boldsymbol{v}\right)_{\mathcal{P}},\left(\boldsymbol{B}^{-1}\boldsymbol{v}\right)_{\overline{\mathcal{P}}}\right)$, with $\left(\boldsymbol{B}^{-1}\boldsymbol{v}\right)_{\mathcal{P}}\in\text{Span}(\boldsymbol{A}_{\cdot\mathcal{P}})$. This decomposition being unique, it follows that $\boldsymbol{v}\in\text{Span}(\boldsymbol{A}_{\cdot\mathcal{P}})$ if and only if $\left(\boldsymbol{B}^{-1}\boldsymbol{v}\right)_{\overline{\mathcal{P}}}=\boldsymbol{0}$.

Elhallaoui et al. (2011) and Raymond et al. (2010b) always assume a complete basis is available because the initial solution comes from a simplex phase I. Here, the computations are minimized by choosing the simplest possible completion, i.e.,

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

As a consequence, the set of compatible columns is determined by forming the matrix $-\boldsymbol{A}_{\overline{\mathcal{P}}\mathcal{P}}\boldsymbol{A}_{\mathcal{P}\mathcal{P}}^{-1}\boldsymbol{A}_{\mathcal{P}\mathcal{N}} + \boldsymbol{A}_{\overline{\mathcal{P}}\mathcal{N}}$. Only for $\boldsymbol{x}_{\mathcal{M}}$ may this calculation be skipped, since those columns are known to be compatible.

In contrast to the identification of compatible columns, the transformation of C-Gni is not necessary for the execution of the algorithm. It is however an essential element of IPS. 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. These constraints are equivalent to $\boldsymbol{A}_{\cdot\mathcal{I}}\boldsymbol{d}_{\mathcal{I}} \in \text{Span}(\boldsymbol{A}_{\cdot\mathcal{P}})$. As stated above, another way of putting this condition is

$$\left(-\boldsymbol{A}_{\overline{\mathcal{P}}\mathcal{P}}\boldsymbol{A}_{\mathcal{P}\mathcal{P}}^{-1}\boldsymbol{A}_{\mathcal{P}\mathcal{I}} + \boldsymbol{A}_{\overline{\mathcal{P}}\mathcal{I}}\right)\boldsymbol{d}_{\mathcal{I}} = \boldsymbol{0}. \tag{5}$$

Moreover, since $\boldsymbol{A}_{\cdot\mathcal{P}}$ is a full-rank matrix, for a given $\boldsymbol{d}_{\mathcal{I}}$ satisfying (5), there is a unique $\boldsymbol{d}_{\mathcal{P}}$ such that $\boldsymbol{A}_{\cdot\mathcal{P}}\boldsymbol{d}_{\mathcal{P}} + \boldsymbol{A}_{\cdot\mathcal{I}}\boldsymbol{d}_{\mathcal{I}} = 0$. This vector is given by $\boldsymbol{d}_{\mathcal{P}} = -\boldsymbol{A}_{\mathcal{P}\mathcal{P}}^{-1}\boldsymbol{A}_{\mathcal{P}\mathcal{I}}\boldsymbol{d}_{\mathcal{I}}$. Denoting $\bar{\boldsymbol{c}}^T = \boldsymbol{c}^T - \boldsymbol{c}_{\mathcal{P}}^T\boldsymbol{A}_{\mathcal{P}\mathcal{P}}^{-1}\boldsymbol{A}_{\mathcal{P}\cdot}$ to extend the usual notation of the reduced cost, the solution of C-Gni is thus equivalently found by solving a smaller LP that involves only the variables of $\boldsymbol{d}_{\mathcal{I}}$:

$$\begin{aligned} z^{\star}_{\text{C-Gni}} = \min_{\boldsymbol{d}_{\mathcal{I}}} \ & \bar{\boldsymbol{c}}_{\mathcal{I}}^T\boldsymbol{d}_{\mathcal{I}} \\ \text{s.t.} \ & \left(-\boldsymbol{A}_{\overline{\mathcal{P}}\mathcal{P}}\boldsymbol{A}_{\mathcal{P}\mathcal{P}}^{-1}\boldsymbol{A}_{\mathcal{P}\mathcal{I}} + \boldsymbol{A}_{\overline{\mathcal{P}}\mathcal{I}}\right)\boldsymbol{d}_{\mathcal{I}} = 0 \\ & \boldsymbol{w}_{\mathcal{I}}^T\boldsymbol{d}_{\mathcal{I}} = 1 \\ & \boldsymbol{d}_{\mathcal{I}_{\mathcal{L}}} \geq 0, \boldsymbol{d}_{\mathcal{I}_{\mathcal{U}}} \leq 0 \end{aligned} \tag{Cr-Gni}$$

This transformation of the complementary problem C-Gni into a smaller problem Cr-Gni is similar to that performed in the reduced-gradient algorithms as presented by Kallio and Porteus (1978). That is, the reduction of a problem with $p + |\mathcal{I}|$ variables to $|\mathcal{I}|$ variables in such a way that $(\boldsymbol{d}_{\mathcal{P}}, \boldsymbol{d}_{\mathcal{I}})$ is easily inferred from $\boldsymbol{d}_{\mathcal{I}}$ resembles the reduced-gradient approach. However, the reduction in the number of constraints presented here is not mentioned in either the theory (see Kallio and Porteus (1978)) or the MINOS implementation

of Wolfe's reduced-gradient (see Murtagh and Saunders (1978)), and neither work suggests adding a normalization constraint that could influence the objective.

Although the density of the constraint matrix changes as we transform C-Gni, in a way that is impossible to predict in the general case, it seems that an important reduction in the computational time could be achieved by considering an LP with fewer variables and constraints. In IPS, R-Gni has p constraints and $p+|\mathcal{C}|$ variables, and Cr-Gni has $m-p$ constraints and $n-p-|\mathcal{C}|$ variables. If the number of simplex pivots correlates with the size of the two problems, one would expect IPS to perform best on problems with $p \approx 0.5m$ and approximately as many compatible as incompatible variables.

Focusing now on the computational burden of the algebraic operations described above, we see that the method used to determine $\mathcal{C}$ is closely linked to the choice of whether or not to transform C-Gni. If $-\boldsymbol{A}_{\overline{\mathcal{P}}\mathcal{P}}\boldsymbol{A}_{\mathcal{P}\mathcal{P}}^{-1}\boldsymbol{A}_{\mathcal{P}\mathcal{N}}+\boldsymbol{A}_{\overline{\mathcal{P}}\mathcal{N}}$ is computed to check the compatibility of the nonbasic variables, there is no other costly operation to execute in order to form Cr-Gni. Therefore, solving Cr-Gni should be faster than solving C-Gni. On the other hand, if a fast method for identifying the compatible columns is available, much of the time spent computing the constraints of Cr-Gni would potentially be saved by keeping the original constraints in C-Gni. With this in mind, Raymond et al. (2010a) recently proposed a stochastic compatibility test that does approximately as many operations as the computation of a reduced cost and identifies all the compatible columns with an extremely small probability of error.

3.3. Setting the Normalization Constraint in the Complementary Problem

The second important implementation choice considered as a part of IPS by Elhallaoui et al. (2011) and Raymond et al. (2010b) is the normalization constraint of C-Gni. For simplicity, all the weights are set to ± 1 in IPS.

To better understand the effect of this choice, one may observe that Cr-Gni could be equivalently solved by removing the normalization constraint and minimizing the normalized criterion $\bar{\boldsymbol{c}}_{\mathcal{I}}^T\boldsymbol{d}_{\mathcal{I}}/\boldsymbol{w}_{\mathcal{I}}^T\boldsymbol{d}_{\mathcal{I}}$. A normalized solution of Cr-Gni would then be obtained as $\boldsymbol{d}_{\mathcal{I}}^\star/\boldsymbol{w}_{\mathcal{I}}^T\boldsymbol{d}_{\mathcal{I}}^\star$. This shows that the normalization constraint actually impacts the criterion guiding the search for an improving compatible combination of columns. For instance, even if it cannot lead to a compatible direction, if the search is restricted to solutions with one nonzero variable, $\bar{c}_i/w_i, i \in \mathcal{I}$ will be minimized. Setting all the weights to ± 1 thus leads to an extension of Dantzig's original pricing criterion (see Dantzig (1955)) to improving

directions with more than one nonzero nonbasic variable. The main advantage of Dantzig's pricing is that the weights are easy to compute, however, there is no reason why they should particularly lead to good directions.

3.4. Solving the Reduced Problem

The practical solution of R-Gni results from the immediate computation of its optimal solution, as indicated by the proposition below.

Proposition 3. R-Gni *admits an optimal solution* $(\boldsymbol{d}^\star_{\mathcal{P}}, \boldsymbol{d}^\star_{\mathcal{C}})$ *such that*

$$\begin{cases} d^\star_j = \dfrac{1}{w_j}, \text{ for some } j \in \arg\min\left(\left\{\dfrac{\bar{c}_i}{w_i} : i \in \mathcal{C}_{\mathcal{L}} \cup \mathcal{C}_{\mathcal{U}}\right\} \cup \left\{-\dfrac{|\bar{c}_i|}{w_i} : i \in \mathcal{M}\right\}\right) \\ d^\star_i = 0 \quad \text{, for all } i \in \mathcal{C} \setminus \{j\} \\ \boldsymbol{d}^\star_{\mathcal{P}} = -\dfrac{1}{w_j}\boldsymbol{A}^{-1}_{\mathcal{P}\mathcal{P}}\boldsymbol{A}_{\cdot j}. \end{cases} \tag{6}$$

Proof Let $z_{\min} = \min\left(\left\{\frac{\bar{c}_i}{w_i} : i \in \mathcal{C}_{\mathcal{L}} \cup \mathcal{C}_{\mathcal{U}}\right\} \cup \left\{-\frac{|\bar{c}_i|}{w_i} : i \in \mathcal{M}\right\}\right)$. For any feasible solution $\boldsymbol{d}$ of R-Gni, $\boldsymbol{d}_{\mathcal{P}} = -\boldsymbol{A}^{-1}_{\mathcal{P}\mathcal{P}}\boldsymbol{A}_{\cdot C}\boldsymbol{d}_{\mathcal{C}}$, so the objective function $\boldsymbol{c}^T_{\mathcal{P}}\boldsymbol{d}_{\mathcal{P}} + \boldsymbol{c}^T_{\mathcal{C}}\boldsymbol{d}_{\mathcal{C}}$ is equivalently written

$$\begin{aligned} \sum_{i\in\mathcal{C}} \bar{c}_i d_i = \sum_{i\in\mathcal{C}} \frac{\bar{c}_i}{w_i} w_i d_i &\geq \sum_{i\in\mathcal{C}_{\mathcal{L}}\cup\mathcal{C}_{\mathcal{U}}} \frac{\bar{c}_i}{w_i} w_i d_i + \sum_{i\in\mathcal{M}} \frac{-|\bar{c}_i|}{w_i} w_i |d_i| \\ &\geq \min\left\{\frac{\bar{c}_i}{w_i} : i \in \mathcal{C}_{\mathcal{L}} \cup \mathcal{C}_{\mathcal{U}}\right\} \sum_{i\in\mathcal{C}_{\mathcal{L}}\cup\mathcal{C}_{\mathcal{U}}} w_i d_i + \min\left\{-\frac{|\bar{c}_i|}{w_i} : i \in \mathcal{M}\right\} \sum_{i\in\mathcal{M}} w_i |d_i| \\ &\geq z_{\min} \times \left(\sum_{i\in\mathcal{C}_{\mathcal{L}}\cup\mathcal{C}_{\mathcal{U}}} w_i d_i + \sum_{i\in\mathcal{M}} w_i |d_i|\right) = z_{\min}. \end{aligned}$$

One can easily verify that a vector $\boldsymbol{d}^\star$ satisfying (6) is a feasible solution of R-Gni whose objective value is equal to $z_{\min}$, so it is optimal. □

The consequence of Proposition 3 is that solving R-Gni and following the optimal solution until a bound is reached is equivalent to pivoting a compatible column into the working basis. As long as the variables of $\boldsymbol{x}_{\mathcal{P}}$ are strictly within their bounds, those pivots are guaranteed to be nondegenerate. In practice, this guarantee may be costly since it implies that variables at one of their bounds must be removed from $\mathcal{P}$ and $\mathcal{C}$ after each pivot. As a consequence, IPS solves R-Gni several times without updating the partition $(\mathcal{P}, \mathcal{C}, \mathcal{I})$ by running the primal simplex on the following *reduced problem* until a stopping criterion is reached:

$$\begin{cases} \min & \boldsymbol{c}^T_{\mathcal{P}}\boldsymbol{x}_{\mathcal{P}} + \boldsymbol{c}^T_{\mathcal{C}}\boldsymbol{x}_{\mathcal{C}} \\ \text{s. t.} & \boldsymbol{A}_{\mathcal{P}\mathcal{P}}\boldsymbol{x}_{\mathcal{P}} + \boldsymbol{A}_{\mathcal{C}\mathcal{P}}\boldsymbol{x}_{\mathcal{C}} = \tilde{\boldsymbol{b}}_{\mathcal{P}} \\ & \boldsymbol{l}_{\mathcal{P}} \leq \boldsymbol{x}_{\mathcal{P}} \leq \boldsymbol{u}_{\mathcal{P}}, \boldsymbol{l}_{\mathcal{C}} \leq \boldsymbol{x}_{\mathcal{C}} \leq \boldsymbol{u}_{\mathcal{C}} \end{cases} \tag{Red}$$

Here $\tilde{\boldsymbol{b}}_{\mathcal{P}} = \boldsymbol{b}_{\mathcal{P}} - \boldsymbol{A}_{\cdot\mathcal{I}_{\mathcal{L}}}\boldsymbol{l}_{\mathcal{I}_{\mathcal{L}}} - \boldsymbol{A}_{\cdot\mathcal{I}_{\mathcal{U}}}\boldsymbol{u}_{\mathcal{I}_{\mathcal{U}}}$ is used instead of $\boldsymbol{b}_{\mathcal{P}}$ to ensure the feasibility of the current solution. The choice of the weights $\boldsymbol{w}_{\mathcal{C}}$ determines the pricing criterion used to solve Red. For instance, if all the weights are set to ± 1, the primal simplex selects entering variables according to Dantzig's criterion.

The stopping criterion is that determined by Raymond et al. (2010b): the simplex is stopped after m iterations if optimality is not reached earlier. In contrast to Raymond et al. (2010b), a new partition is not built every time m consecutive pivots are performed because identifying the compatible variables can be costly. Instead, Red is built anew only if at least 10% of the variables in $\mathcal{P}$ are equal to one of their bounds and if 30% of $\mathcal{P}$ has changed since the last update of $(\mathcal{P},\mathcal{C},\mathcal{I})$. The first criterion ensures that no update of Red is performed provided the degeneracy remains low enough. The second guarantees that some progress is made between two consecutive manipulations of the constraints.

3.5. Solving the Complementary Problem

Once an optimal solution of Red has been found, new improving directions are found or optimality is proved by solving Cr-Gni. Theoretically, each time that Cr-Gni is solved it provides a feasible descending direction. However, this property requires that the partition $(\mathcal{P},\mathcal{C},\mathcal{I})$ is updated each time. Furthermore, following the improving direction found by Cr-Gni usually results in only a small modification of the current solution and the associated Red. For these reasons, the strict theoretical framework described in Algorithm 2 is usually inefficient in practice. As stated at the end of Subsection 3.4, the decomposition is thus updated only when the solution is degenerate enough and $\mathcal{P}$ has changed significantly since the last update or $z^{\star}_{\text{C-Gni}} = 0$. Following the procedure of Raymond et al. (2010b), we then solve the complementary problem several times in a row. Each time we remove the variables that appear in the resulting direction from the problem and solve Cr-Gni again to find a new direction involving different nonbasic variables. These directions are not directly followed; the nonzero variables are instead included in $\mathcal{C}$ together with the other variables belonging to $\text{Span}\,(\boldsymbol{A}_{\cdot\mathcal{P}\cup\mathcal{C}})$. This is done to grant the simplex more flexibility in choosing the order of the pivots when solving the resulting Red. In our tests, complementary problems were solved until at least 10% of the columns of $\mathcal{I}$ were selected to be appended to the reduced problem.

As in Raymond et al. (2010b), Cr-Gni is solved with the dual simplex algorithm but no warm start is described. An improvement is added to ensure that this process starts from

a dual feasible solution. With that in mind, consider the dual formulation of Cr-Gni with slack variables $\boldsymbol{s}$:

$$\begin{aligned} z^{\star}_{\text{C-Gni}} = \max_{(\boldsymbol{\pi},y)} \; & y \\ \text{s.t.} \; & w_i y + \boldsymbol{\pi}^T \left(-\boldsymbol{A}_{\overline{\mathcal{P}}\mathcal{P}} \boldsymbol{A}_{\mathcal{P}\mathcal{P}}^{-1} \boldsymbol{A}_{\mathcal{P}i} + \boldsymbol{A}_{\overline{\mathcal{P}}i}\right) + s_i = \bar{c}_i, \quad \forall i \in \mathcal{I}_{\mathcal{L}} \\ & w_i y + \boldsymbol{\pi}^T \left(-\boldsymbol{A}_{\overline{\mathcal{P}}\mathcal{P}} \boldsymbol{A}_{\mathcal{P}\mathcal{P}}^{-1} \boldsymbol{A}_{\mathcal{P}i} + \boldsymbol{A}_{\overline{\mathcal{P}}i}\right) - s_i = \bar{c}_i, \quad \forall i \in \mathcal{I}_{\mathcal{U}} \\ & \boldsymbol{s} \geq \boldsymbol{0}, \boldsymbol{\pi} \in \mathbb{R}^{m-p}, y \in \mathbb{R} \end{aligned} \tag{7}$$

With $j \in \arg\min\{\bar{c}_i/w_i : i \in \mathcal{I}\}$, a feasible solution of (7) is obtained by setting $\boldsymbol{\pi} = 0$, $y = \bar{c}_j/w_j$, $s_j = 0$, and $s_i = \pm(\bar{c}_i - \bar{c}_j \frac{w_i}{w_j}) \, (\geq 0), \forall i \in \mathcal{I} \setminus \{j\}$. The associated basis is made up of y and $\{s_i\}_{\{i \in \mathcal{I} \setminus \{j\}\}}$, while $\boldsymbol{\pi}$ and s_j are nonbasic. Clearly, this solution is not optimal since the complementarity conditions would lead to only one nonzero variable in $\boldsymbol{d}_{\mathcal{I}}$, which contradicts the definition of an incompatible variable. On the other hand, giving this solution to a LP solver helps to avoid a potentially time-consuming phase I.

Once an optimal solution $\boldsymbol{d}^{\star}_{\mathcal{I}}$ of Cr-Gni has been found, all the variables such that $d^{\star}_i \neq 0, i \in \mathcal{I}$, are removed from Cr-Gni, which is equivalent to removing the associated constraints and slack variables in (7). As a consequence, the solution remains dual feasible and can be used to efficiently warm-start the solution process for this smaller Cr-Gni.

3.6. Summary of the Algorithm

The choices and adaptations that are included in IPS to obtain an efficient practical implementation are summarized in Algorithm 3. The set $\mathcal{P}$ is not necessarily updated after each modification of the solution. Instead, we denote by $\mathcal{P}^+$ the set indexing the variables strictly within their bounds. Immediately after building $(\mathcal{P},\mathcal{C},\mathcal{I})$ from scratch (step 5), $\mathcal{P} = \mathcal{P}^+$, but this may change after pivots are performed. The reduced and complementary problems, Red and Cr-Gni, correspond to the current partition $(\mathcal{P},\mathcal{C},\mathcal{I})$. The updates of $(\mathcal{P},\mathcal{C},\mathcal{I})$ at step 5 thus correspond to a decrease in the size of Red while those at step 17 lead to an increase.

4. Extended Computational Analysis of IPS

4.1. Solution of the LPs

The optimization library cplex 12.4[1] is called for the solution of every LP. The initial primal feasible solution is generated through a simplex phase I; the primal simplex is used

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

Algorithm 3: Improved primal simplex with upper and lower bounds

Input: A linear program P, $\boldsymbol{x}^0$ a basic feasible solution of P.

Output: $\boldsymbol{x}$, an optimal solution of P.

$\boldsymbol{x} \leftarrow \boldsymbol{x}^0$; $\mathcal{P} \leftarrow \{j : x_j^0 \text{ is basic}\}$; $\mathcal{P}^+ \leftarrow \{j : l_j < x_j^0 < u_j\}$; $\mathcal{P}_0 \leftarrow \emptyset$; *optimal* $\leftarrow$ *false*;
while *optimal* = *false* **do**
  while $\boldsymbol{x}$ *is not optimal for* RED **do**
    if $|\mathcal{P}^+| \leq 0.9\,|\mathcal{P}|$ *and* $|\mathcal{P}_0 \cap \mathcal{P}^+| \leq 0.7\,|\mathcal{P}_0|$ **then**
      Build the partition $(\mathcal{P},\mathcal{C},\mathcal{I})$ corresponding to $\boldsymbol{x}$; $\mathcal{P}_0 \leftarrow \mathcal{P}$;
    Solve RED for m iterations or until optimality: $\boldsymbol{x} \leftarrow$ the current solution;
    $\mathcal{P}^+ \leftarrow \{j : l_j < x_j < u_j\}$;
  Solve CR-GNI: $\boldsymbol{d}_{\mathcal{I}}^\star \leftarrow$ 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 $\boldsymbol{d}_{\mathcal{I}}^\star$ and $z^\star_{\text{CR-GNI}}$;
      $\mathcal{I}^+ \leftarrow \mathcal{I}^+ \cup \{i \in \mathcal{I} : d_i^\star > 0\}$;
    Complete $\mathcal{P}$ with indices of $\mathcal{I}^+$ to get a basis of Span$(\boldsymbol{A}_{\cdot\mathcal{P}\cup\mathcal{I}^+})$; update $(\mathcal{C},\mathcal{I})$;

to solve RED while the dual simplex is used for CR-GNI. IPS is then evaluated by comparing its performance with a direct solution of the complete problem P by the primal simplex of CPLEX, starting from the same initial primal feasible solution. For convenience, this direct primal simplex approach is referred to as DPS in the following discussion. Moreover, we performed the CPLEX presolve initially and ran the simplex phase I, IPS and DPS on the presolved instances. This way, the two algorithms take advantage of both the presolve and the warm start.

The default options of CPLEX are used for all the parameters except the pricing method. With default pricing, CPLEX automatically switches between the different criteria implemented for the primal simplex, depending on indicators that are unfortunately not documented or available to the user. There are several disadvantages to this automatic pricing. First and foremost, iteratively starting and stopping CPLEX has a strong undesirable impact

on the choices it makes with regards to the selected criterion. As a consequence, the differences between the DPS and IPS performance could originate from the differing pricing methods, which would make interpretation difficult. Second, the choice of the pricing has important consequences on the number of pivots performed while solving an LP and the computational time for each pivot, as illustrated in Table 1. The automatic pricing would thus invalidate any comparison of the number of pivots. We therefore chose a specific pricing method for all the calls to CPLEX, both in DPS and IPS. To choose the method, we tested the three usually most efficient pricing methods implemented in CPLEX on the benchmark detailed in the next section. These methods are the automatic pricing, the devex method of Harris (1973), and an approximate version of the steepest-edge criterion first described by Goldfarb and Reid (1977). These methods respectively correspond to the values 0, 1, and 3 for the CPX_PARAM_PRIIND parameter. Table 1 shows the average runtime and the number of pivots for each problem family. For an easier comparison, we present the results for devex and steepest-edge as geometric means of the ratios relative to the default option. The results show that the approximate steepest-edge is neither systematically better nor worse than the automatic pricing, and so we used it in all the tests.

Table 1 Comparison of three pricing criteria in cplex

	default		devex (ratios)		steepest-edge (ratios)	
Instance	pivots	cpu	pivots	cpu	pivots	cpu
UBFA	128165	192.00 s	1.08	0.65	0.31	0.25
UFL	52926	17.56 s	1.63	0.94	0.76	0.74
VCS	169552	62.78 s	1.05	3.61	0.28	1.51
Mittelmann	672386	255.92 s	0.62	1.67	0.21	1.23

4.2. Description of the Benchmark

The performance of IPS was tested on five families of problems. The first three sets of instances correspond to specific types of problems: combined fleet assignment and aircraft routing problems (UBFA), uncapacitated warehouse location problems (UFL), and simultaneous vehicle and crew scheduling problems (VCS). Instances of these problems were used by Elhallaoui et al. (2011) because they exhibit significant degeneracy. We made two noteworthy modifications to this benchmark. The ten VCS instances of Elhallaoui et al.

Table 2 Characteristics of the benchmark

	size of P			DPS solve			degeneracy	
Instance	m: rows	n: cols	ρ_A	$\text{cpu}_{\text{p}_\text{I}}$	piv	cpu	var	piv
UBFA1	5182	23650	2.5E-3	23.64	11129	8.78	74.0%	65.2%
UBFA2	5182	23990	2.5E-3	30.09	15566	13.05	74.5%	66.9%
UBFA3	5182	24282	2.5E-3	35.47	67792	93.72	76.2%	68.5%
UBFA4	5182	24517	2.5E-3	28.60	142550	255.36	76.5%	73.2%
UBFA5	5182	24875	2.5E-3	31.03	53591	73.42	76.9%	85.8%
UFL1	7965	7965	2.4E-4	0.02	20327	5.31	57.3%	45.6%
UFL2	10440	10440	1.9E-4	0.02	36072	8.94	69.7%	60.2%
UFL3	15476	15476	1.3E-4	0.03	54069	17.57	75.1%	61.3%
UFL4	20534	20534	9.6E-5	0.04	60587	23.72	76.7%	71.4%
UFL5	25931	25931	7.7E-5	0.05	58382	23.21	80.2%	84.3%
VCS1	2082	10259	1.5E-2	7.35	18598	16.51	44.6%	70.7%
VCS2	2082	10065	1.5E-2	7.72	18631	16.64	44.1%	65.3%
VCS3	2081	26260	1.6E-2	0.83	85014	149.58	62.3%	47.9%
VCS4	1199	130902	1.7E-2	99.47	12327	62.78	53.1%	64.2%
VCS5	1600	551346	1.2E-2	2757.59	26126	591.57	58.2%	68.9%
dano3	3150	13837	1.6E-3	0.15	13392	5.01	33.5%	56.9%
dbic1	33666	140273	1.4E-4	0.30	19943	7.71	88.7%	85.0%
dfl001	3792	8984	8.8E-4	1.98	21059	6.51	46.8%	77.0%
fome12	15168	35936	2.2E-4	2.58	85321	51.23	48.5%	76.2%
fome13	30336	71872	1.1E-4	2.58	155263	146.29	48.9%	71.5%
l30	2698	15363	1.2E-3	6.18	281111	246.24	22.9%	96.5%
lp22	2872	8693	1.9E-3	3.21	28187	14.03	30.1%	60.9%
mod2	23828	23904	1.2E-4	5.34	38619	31.62	16.2%	13.1%
neos1'	1892	131581	1.9E-3	16.50	34064	116.10	82.3%	57.1%
neos2'	1558	131875	2.7E-3	18.47	33565	114.37	82.4%	20.8%
ns1688926	24576	16489	1.1E-3	0.26	17581	57.50	43.6%	69.6%
nsct2	7778	11297	4.2E-3	0.12	8411	0.40	89.6%	62.2%
nug08-3rd	18269	20447	2.5E-4	642.70	462736	9649.38	99.5%	99.9%
rail4284	4176	1090526	2.4E-3	3.28	89959	2568.54	26.2%	38.5%
rlfprim'	4046	40340	9.8E-4	0.05	3198	1.06	96.4%	99.9%
stat96v1	4502	187604	6.6E-4	0.34	40208	171.55	39.4%	75.9%
world	23752	25702	1.2E-4	2.66	44396	38.94	17.7%	13.6%

(2011) all had 2084 constraints and about 6000 to 10000 variables. We kept only the largest two, and we added three others involving two to fifty times more variables to observe the sensitivity of the algorithm to variations in the number of variables. We derived the UBFA instances from the five largest fleet assignment instances of Elhallaoui et al. (2011) by explicitly adding upper bounds on the variables. The set-partitioning constraints implicitly constrain the variables to be less than or equal to one; we added the upper bound $x_i \leq 1$ to every variable x_i. Additionnally, presolve was switched off for these instances to prevent it from removing these implied bounds. We emphasize that we did not modify the original fleet assignment instances to improve the performance of the algorithm with respect to the

results of Elhallaoui et al. (2011). Our goal is only to build quickly a test set for a first evaluation of the algorithm with upper and lower bounds.

The instances used by Elhallaoui et al. (2011) and Raymond et al. (2010b) share common features that motivated their choice. They are all highly degenerate, their sizes are reasonable, the variables are all bounded below (by 0) but not above, and they include similar types of constraints. For instance, the UBFA, and VCS problems all contain a majority of set-partitioning constraints. Although we made some changes to the benchmark by considering larger instances and adding finite upper bounds to the fleet-assignment instances, it is difficult to make a thorough analysis of the strengths and weaknesses of IPS for such a homogeneous benchmark. We therefore added instances selected from Mittelmann's benchmark[2], which was built to enable a rigorous evaluation of the commercial and open implementations of the most efficient algorithms for LP. To keep the size of the benchmark reasonable, we only kept the instances with less than 50000 constraints. We also considered the dual form of the instances containing significantly more constraints than variables when the number of constraints of the dual LP is less than 50000 constraints. Even with this restriction, some selected instances are large enough to exhibit the difficulties that IPS may encounter when the number of rows increases. Moreover, in Algorithm 3, the partition is not built as long as less than 10% basic variables are degenerate. This threshold is never reached when solving gen4, qap12, qap15, nug15 and self, so IPS is equivalent to DPS. As a consequence, we do not display the results for these instances. Testing IPS on the remaining instances should help to confirm its strengths and identify its limits.

Table 2 summarizes the characteristics of the instances that we used. It focuses on three aspects of the instances: their sizes, the performance of DPS, and indicators of degeneracy. Since the algorithms are run after the presolve, the characteristics given in the table refer to the presolved instances (except for UBFA1–5). The header $\rho_{\boldsymbol{A}}$ stands for the density of the original constraint matrix $\boldsymbol{A}$, and $\text{cpu}_{\text{p}_\text{I}}$ is the time spent in the simplex phase I. To characterize the degeneracy during the solution of the instances, we recorded both the average number of degenerate variables and the number of degenerate pivots. The Mittelmann instances for which we solved the dual form are indicated with a $'$ symbol.

[2] These instances are available online: `http://plato.asu.edu/ftp/lpcom.html`

4.3. Experimental Results

The tests were all performed on an OpenSuse operating system with an Intel(R) Core(TM) i7-3770 CPU @ 3.40 GHz processor.

IPS as summarized in Algorithm 3 was called on the benchmark to obtain the results in Table 3. These results are relevant to a first analysis of the IPS performance. The headers are partitioned into three types of information. The first two columns record the number of times the partition $(\mathcal{P},\mathcal{C},\mathcal{I})$ was updated ("$\mathcal{PCI}$") and the number of times the reduced problem was expanded by solving Cr-Gni several times ("EXP"). The following columns give the proportion of the runtime spent in computing the partitions $(\mathcal{P},\mathcal{C},\mathcal{I})$, solving Red, and solving Cr-Gni, as well as the total runtime of IPS. For an easier comparison, the last four columns contain the ratios of the performance measures of DPS relative to those of IPS, thus indicating the improvement factor of IPS. The headers "piv_{Red}" and "deg piv" respectively stand for the total number of primal simplex pivots and the proportion of degenerate pivots, and "cpu" stands for the total runtime. The quantity in the "cpu_{Red}" column is the ratio of the time spent solving Red relative to the DPS runtime. This column gives a measure of the improvement that could be expected for IPS if $(\mathcal{P},\mathcal{C},\mathcal{I})$ was updated rapidly and the solution of C-Gni was fast even if the problem is not transformed (since the transformation of C-Gni comes with costly updates of $(\mathcal{P},\mathcal{C},\mathcal{I})$). Geometric averages are provided for the improvement factors of the first three types of problems, but given the heterogeneity of the Mittelmann instances it did not seem relevant to base an interpretation on average values. The analysis of the Mittelmann instances is instead done on a case-by-case basis.

The solution was on average 4 times faster with IPS than with DPS for the VCS instances. Similar results were observed by Elhallaoui et al. (2011) and Raymond et al. (2010b) for instances whose sizes are comparable with those of VCS1 and VCS2. The interesting point is that IPS remains efficient on VCS3, VCS4, and VCS5, where the number of variables is two to fifty times larger. As expected, the total number of pivots and the proportion of degenerate pivots are lower for IPS than for DPS, and an important indicator of efficiency is that the time spent in transforming and solving Cr-Gni remains a small part of the total runtime. The instance VCS4 seems to be an exception, but this is because that IPS performed many fewer pivots in Red, thus giving more importance to the partition updates and Cr-Gni.

Table 3 Evaluation of IPS against cplex

	phase count		cpu allocation				improvement factors			
	$\mathcal{PCI}$	EXP	$\mathcal{PCI}$	RED	CR-GNI	cpu	piv_{RED}	deg piv	cpu_{RED}	cpu
VCS1	4	10	3.3%	82.6%	12.9%	6.15	1.49	2.17	3.25	2.68
VCS2	3	10	2.7%	85.6%	10.6%	6.02	1.55	1.47	3.23	2.76
VCS3	3	12	3.0%	56.6%	39.3%	15.97	5.10	1.23	16.56	9.37
VCS4	1	4	3.0%	85.2%	10.4%	18.16	1.07	1.18	4.06	3.46
VCS5	1	4	1.7%	93.4%	4.3%	128.23	1.62	1.23	4.94	4.61
Average							1.83	1.42	5.11	4.06
UBFA1	1	7	17.4%	45.3%	34.3%	6.08 s	1.12	1.90	3.19	1.44
UBFA2	1	8	16.5%	44.4%	35.6%	6.47 s	1.49	1.86	4.54	2.02
UBFA3	1	8	16.0%	48.5%	32.3%	6.72 s	5.94	1.56	28.74	13.94
UBFA4	1	8	14.6%	43.0%	39.4%	7.77 s	11.41	1.97	76.43	32.86
UBFA5	1	10	14.9%	49.3%	32.5%	7.26 s	4.26	1.89	20.53	10.12
Average							3.44	1.83	14.55	6.70
UFL1	3	15	18.4%	57.0%	21.3%	3.79 s	1.76	1.43	2.46	1.40
UFL2	3	13	25.9%	38.4%	31.8%	4.37 s	3.21	1.47	5.32	2.04
UFL3	4	12	32.6%	37.4%	26.8%	9.00 s	2.80	1.56	5.22	1.95
UFL4	3	11	32.2%	33.7%	31.4%	14.23 s	2.51	2.32	4.94	1.67
UFL5	4	10	43.8%	19.8%	33.9%	18.44 s	2.82	1.78	6.36	1.26
Average							2.57	1.68	4.64	1.64
dano3	2	6	6.7%	73.8%	17.7%	3.66	1.16	1.61	1.86	1.37
dbic1	1	4	29.6%	4.3%	64.2%	210.63	0.93	1.16	0.86	0.04
dfl001	2	10	8.0%	58.5%	31.7%	5.43	1.23	1.53	2.05	1.20
fome12	3	14	13.6%	29.7%	55.2%	104.6	0.98	2.08	1.65	0.49
fome13	3	14	12.8%	22.0%	64.0%	455.82	0.90	1.36	1.46	0.32
l30	2	3	34.6%	8.3%	56.3%	151.37	12.29	1.32	19.58	1.63
lp22	6	9	5.5%	84.5%	9.4%	11.29	1.01	1.78	1.47	1.24
mod2	1	9	3.5%	93.6%	2.6%	38.94	0.80	1.37	0.87	0.81
neos1'	3	6	45.2%	3.7%	42.6%	6.87	16.57	0.66	456.74	16.90
neos2'	2	4	44.3%	18.4%	30.6%	5.4	7.15	0.22	115.23	21.18
ns1688926	1	1	52.3%	41.6%	6.0%	70.88	1.76	1.46	1.95	0.81
nsct2	2	7	26.3%	2.0%	69.3%	8.5	1.54	0.68	2.43	0.05
nug08-3rd	3	3	0.2%	0.3%	99.5%	5810.19	272.52	1.00	553.59	1.66
rail4284	1	9	1.3%	84.3%	13.8%	1386.36	1.07	0.73	2.20	1.85
rlfprim'	2	1	56.2%	0.2%	42.9%	4.87	1599.00	∞	120.92	0.22
stat96v1	3	3	14.9%	45.4%	39.5%	126.5	2.33	1.12	3.00	1.36
world	1	10	3.2%	93.5%	3.0%	44.31	0.84	1.17	0.94	0.88

Referring to Table 2, it appears that around 75% of the basic variables were degenerate for the UBFA instances. About one third of these variables are at their upper bounds ($x_i = 1$) while the rest are at their lower bounds ($x_i = 0$). Because of this high number of degenerate variables, 65% to 80% of the pivots of DPS are degenerate, and there is a great variability in the number of pivots although the instances all derive from the same problem and have similar sizes. The average improvement by a factor of 6.7 for this class of problem reflects a stabilization of the solution process for these highly degenerate problems: as expected, the IPS runtimes are close for these instances, ranging from 6.1 s to 7.8 s, and

they globally increase with the instance sizes. These positive results validate the potential usefulness of IPS for degenerate problems with bounded variables.

The UFL instances clearly illustrate the impact of the partition updates on the overall execution of IPS. Although IPS achieves a considerable reduction in the number of pivots, leading to an average 2.6 improvement factor and a proportion of degenerate pivots that is 1.7 times smaller than that for DPS, the total improvement is not as good as that for the other families of problems. This is mostly due to the computational time spent in the update of $(\mathcal{P},\mathcal{C},\mathcal{I})$ (see column "$\mathcal{PCI}$"), which seems to grow much faster than the time spent solving RED as the number of constraints increases. As a consequence, the column "cpu_{RED}" shows a large improvement factor (4.6 on average) over DPS. This shows the potential for improvement in IPS if the partition can be updated quickly and the ensuing transformation of C-GNI avoided in such cases.

In contrast to the previously examined families of instances, the performance of IPS is much less homogeneous for the Mittelmann instances. A significant improvement (greater than 20%) was achieved for 9 of the 17 instances (dano3, dfl001, l30, lp22, neos1', neos2', nug08-3rd, rail4284, stat96v1), the computational time was significantly worse for 7 instances (dbic1, fome12, fome13, mod2, ns1688926, nsct2, rlfprim'), and it was similar for the remaining instance (world).

Starting with the largest improvements, we see that the success of IPS on the two similar instances neos1' and neos2' is due to the large proportion of degenerate variables (82%). Because of this, RED is small, hence it is solved quickly. An impressive improvement factor for the solution of RED alone was also recorded for nug08-3rd and l30. For these instances, a massive proportion of degenerate pivots (96.5% to 99.9%) are performed when executing the primal simplex in DPS (see Table 2). With IPS, a large number of degenerate pivots is avoided by focusing on the compatible variables and on the improving directions identified by CR-GNI. These results are consistent with the purpose of IPS.

The other five instances for which IPS performed significantly better than DPS (dano3, dfl001, lp22, rail4284, stat96v1) share several attributes: they have comparable constraint matrix densities and numbers of constraints, and the proportion of degenerate variables ranges from 26% to 46%. Since the computational time spent updating $(\mathcal{P},\mathcal{C},\mathcal{I})$ depends on the number of constraints and the number of degenerate variables, it is logical that it is responsible for only a small part of the total computational time. On the other hand,

although the overall cpu improvement factors are close (ranging from 1.2 to 1.9), the solution of Cr-Gni represents a great proportion of the total runtime for dfl001 and stat96v1. As a consequence, a significant gain could be achieved if the solution of Cr-Gni could be sped up for these instances; an improvement factor ranging from 2 to 3 was recorded for the solution of Red alone.

The remaining eight Mittelmann instances did not lead to a significant improvement. The possible identification of probable causes is supported by Table 4. This table is restricted to these eight instances. The bold font highlights the information most relevant to explaining the low improvement factors, and the instances with similar features are grouped in adjacent rows. The columns related to degeneracy and the improvement factors reproduce the corresponding columns of Tables 2 and 3. The second group of columns focuses on the partition updates. To better emphasize the comparison with DPS, the computational time spent in the updates is divided by the DPS runtime. The second column ("$|\mathcal{C}|\,/\,|\mathcal{P}|$") displays the maximum value of the ratio of the cardinality of $\mathcal{C}$ and $\mathcal{P}$ found after the partition updates. This value is an indicator of the potential progress made when solving Red: if it is large, there is a greater probability that many cheap pivots are performed. The third group of columns focuses on the solution of Cr-Gni. The first column indicates the impact of the transformation of C-Gni on the density of its constraint matrix. The second column shows the Cr-Gni computational time divided by the DPS runtime.

Table 4 Focus on the unimproved instances

	degeneracy		updates of $(\mathcal{P},\mathcal{C},\mathcal{I})$		solve Cr-Gni		improvement factors	
	var	piv	$\frac{\text{cpu}}{\text{cpu}_{\text{DPS}}}$	$\lvert\mathcal{C}\rvert\,/\,\lvert\mathcal{P}\rvert$	$\frac{\rho_{\text{Cr-Gni}}}{\rho_{\boldsymbol{A}}}$	$\frac{\text{cpu}}{\text{cpu}_{\text{DPS}}}$	cpu_{Red}	cpu
mod2	**16.2%**	**13.1%**	4.3%	92.5%	5.1	3.2%	0.87	0.81
world	**17.7%**	**13.6%**	3.6%	98.8%	4.6	3.4%	0.94	0.88
dbic1	88.7%	85.0%	**808.7%**	**6.2%**	4.5	**1754.7%**	0.86	0.04
fome12	48.5%	76.2%	27.8%	**43.2%**	1.6	**112.6%**	1.65	0.49
fome13	48.9%	71.5%	39.9%	**45.1%**	1.6	**199.3%**	1.46	0.32
ns1688926	43.6%	69.6%	**64.5%**	95.1%	2.0	7.4%	1.95	0.81
nsct2	89.6%	62.2%	**555.8%**	613.0%	1.0	**1461.5%**	2.43	0.05
rlfprim'	96.4%	99.9%	**257.5%**	0.0%	1.6	**197.2%**	120.92	0.22

The first group of instances, mod2 and world, exhibit low proportions of degenerate variables and pivots when solved by DPS. Since the main benefits of IPS are the opportunity to solve a reduced problem and to perform fewer degenerate pivots, these two instances

leave little room for improvement. The IPS runtime is larger than that for DPS, and so is the time spent solving RED. It appears that IPS is not appropriate for such instances. A simple adaptation would be to wait for an important proportion of degenerate pivots to be performed before starting the algorithm. With such a modification, IPS would be equivalent to DPS on such instances.

Regarding the second group (dbic1, fome12 and fome13), the most probable cause of the mediocre performance of IPS is the number of compatible variables after each reduction. The direct consequence is that after the reduction, a large majority of the variables in RED belong to the working basis, leaving little hope for important progress toward optimality. This also has a direct impact on the time spent solving the complementary problem since many variables have to be returned to RED before it includes a sufficient number of variables. A simple adaptation would be to estimate the number of compatible variables before starting IPS. However, that is currently costly since it requires the computation of $-\boldsymbol{A}_{\overline{\mathcal{P}}\mathcal{P}}\boldsymbol{A}_{\mathcal{P}\mathcal{P}}^{-1}\boldsymbol{A}_{\mathcal{P}\mathcal{N}}+\boldsymbol{A}_{\overline{\mathcal{P}}\mathcal{N}}$, so a faster method would need to be used.

The two instances ns1688926 and dbic1 illustrate the difficulties that IPS may encounter when the size of the LP increases. The partition update may then take more time than the complete solution with the primal simplex of DPS. Moreover, the complementary problem may need to be solved a large number of times before selecting 10% of the incompatible variables, thus leading to a large runtime for CR-GNI. The same behavior was also observed for the two smaller instances nsct2 and rlfprim'. Their particularity is that their solution times with DPS is much smaller than for the other Mittelmann instances with comparable sizes. As a consequence, the overhead generated by the dynamic reduction is too large for IPS to compete with the primal simplex. Once again, the only possible fix for such situations would be to run IPS with fast partition updates and no transformation of C-GNI.

REMARK 2. rlfprim' is a special case because the initial feasible solution is optimal. With DPS, 3198 degenerate pivots are performed before proving optimality, whereas IPS solves CR-GNI once before declaring that the solution is optimal.

Finally, it appears that the matrix of CR-GNI is up to 5 times denser than $\boldsymbol{A}$ for these eight instances. This certainly affects the solution of CR-GNI and is another motivation for IPS not to resort to the transformation of $\boldsymbol{A}$.

REMARK 3. Overall, we observed that the transformation of C-GNI leads to a less dense matrix only for the VCS, and UBFA instances, which concurs with the global deduction

that IPS has a special affinity with certain classes of problems, such as set partitioning problems.

5. Conclusion

We have developed a new decomposition-based primal algorithm for degenerate LPs. The decomposition relies on the partition of the nonbasic variables into compatible ($\mathcal{C}$) and incompatible variables ($\mathcal{I}$), the first group containing the variables that can be pivoted into the working basis $\mathcal{P}$ with a strict improvement in the objective value. This separates the overall improvement problem, Gni, into two smaller problems, R-Gni and C-Gni, respectively focusing on $\mathcal{C}$ and $\mathcal{I}$. We have shown that solving R-Gni and C-Gni is equivalent to solving Gni, and we have proposed a decomposition procedure to find the optimum of the LP.

Among the multiple possible implementations of this generic framework is the IPS algorithm, first described by Elhallaoui et al. (2011). The following developments focus on the choices to be made and the issues to be addressed when moving from the theoretical procedure to IPS. Some new insight into the solution of R-Gni and C-Gni results from this approach that considers IPS as a particular implementation of a much more general algorithm. We make a link between the normalization weights and the pricing criterion; we explain how the algorithm can start from any feasible solution, whether or not it is basic; and we highlight the importance of warm-starting the solution of C-Gni. It appears that the most important choices relate to the identification of the compatible variables, the transformation of the constraint matrix in C-Gni, and the choice of the weights appearing in its normalization constraint. For instance, the first two choices involve costly algebraic operations, while the normalization weights are simply set to ± 1, thus extending the Dantzig pricing criterion to improving directions with multiple nonbasic nonzero variables.

Finally, we conducted experiments on a diversified version of the instances of Elhallaoui et al. (2011) extended with fifteen data sets from Mittelmann's benchmark. IPS outperformed direct solution with the primal simplex on several classes of problems including set partitioning, set covering, and warehouse location problems. It also achieved impressive gains for some very degenerate instances from Mittelmann's benchmark. On the other hand, it gave no improvement on half of Mittelmann's benchmark. A thorough analysis of

these results identified criteria that should be checked before switching from the standard simplex algorithm to IPS. It is necessary to observe a sufficiently large number of degenerate variables and pivots, otherwise there is no reason why IPS should perform better than the primal simplex. Moreover, the number of compatible variables must be large enough, or solving RED will not make significant progress toward optimality. As for the pricing criteria, perturbations, bound shifting, and other techniques included in efficient implementations of the simplex algorithm, IPS should not used blindly but only when certain criteria are met.

We also demonstrated the limits of the current update of the partition $(\mathcal{P}, \mathcal{C}, \mathcal{I})$, the most important one being that it may be time consuming. This indicates the necessity for an implementation of IPS that would update $(\mathcal{P}, \mathcal{C}, \mathcal{I})$ quickly and solve C-GNI efficiently even without transformation of its constraint matrix. The positive-edge criterion described by Raymond et al. (2010a) could be considered since it is a fast method for the identification of compatible variables. Moreover, we noticed that although the transformation leads to a much smaller form of C-GNI, it may also increase the density of the constraint matrix. Further experimentation will be needed to get more insight into this issue.

Our theoretical and experimental analyses suggest numerous interesting directions for further research. For instance, since solving C-GNI or R-GNI does not always provide feasible improving directions in IPS, the important nondegeneracy property of the generic theoretical scheme is lost for the sake of practical efficiency. As a consequence, it is also interesting to interpret IPS as an analytical—as opposed to systematic—partial pricing strategy for the primal simplex algorithm. Let $\mathcal{J} \subset \mathcal{N}$ be the set of nonbasic variables that are considered during the pricing. This set is initialized with the set of compatible variables $\mathcal{C}$. Later on, $\mathcal{J}$ goes through several consecutive phases of expansions and reductions respectively triggered by insufficient progress or problematic degeneracy. The reduction is done by keeping in $\mathcal{P}$ only the variables strictly within their bounds and updating $\mathcal{C}$ accordingly. The expansion is performed by solving a sequence of complementary problems, as described in Subsection 3.5. This expansion is an unusually complex procedure for a partial pricing, which is why we qualified it as "analytical." This interpretation could certainly be fruitful for further improvements of IPS.

McCormick and Shioura (2000) proved that a particular choice of the weights appearing in the normalization constraints makes the primal improving procedure polynomial

if the improvement problem is considered as an oracle. This result underlines the special attention that should be given to this constraint in future developments. With additional calculations, the weights could be modified to emulate the steepest-edge rule of Goldfarb and Reid (1977), the devex of Harris (1973), or any other normalized pricing rule. One may also devise another normalization resulting in a faster solution of C-Gni.

From a more practical point of view, using an open optimization library would be valuable. As in Towhidi et al. (2014), the integration of IPS as an internal procedure of the COIN-OR CLP solver would have several benefits. The extended control over the simplex algorithm would allow us to adapt the choices made during the solution process to the particularities of IPS. Moreover, with complete information on the simplex algorithm available, it would be possible to fill some gaps in the interpretation of the performance of IPS.

Finally, we have focused on the primal version of the simplex algorithm. However, better results are often reported for the dual simplex, and it is also a powerful reoptimization tool used for instance in mixed integer programming to explore the branch-and-bound tree. For these reasons, it would be interesting to develop a dual version of IPS to take advantage of dual degeneracy when solving LPs with the dual simplex.

References

Benichou, M., J. M. Gauthier, G. Hentges, G. Ribiere. 1977. The efficient solution of large-scale linear programming problems: Some algorithmic techniques and computational results. *Mathematical Programming* **13** 280–322.

Dantzig, G. B. 1955. The general simplex method for minimizing a linear form under inequality constraints. *Pacific Journal of Mathematics* **5** 183–195.

Elhallaoui, I., A. Metrane, G. Desaulniers, F. Soumis. 2011. An improved primal simplex algorithm for degenerate linear programs. *INFORMS Journal on Computing* **23** 569–577.

Elhallaoui, Issmail, Daniel Villeneuve, Franois Soumis, Guy Desaulniers. 2005. Dynamic aggregation of set-partitioning constraints in column generation. *Operations Research* **53** 632–645.

Gill, P. E., W. Murray, M. A. Saunders, M. H. Wright. 1989. A practical anti-cycling procedure for linearly constrained optimization. *Mathematical Programming* **45** 437–474. doi:10.1007/BF01589114.

Goldfarb, D., J. Reid. 1977. A practicable steepest-edge simplex algorithm. *Mathematical Programming* **12** 361–371.

Greenberg, H. J. 1978. *Design and Implementation of Optimization Software*, chap. Pivot selection tactics. Sijthoff & Noordhoff, 109–142.

Harris, P. M. J. 1973. Pivot selection method of the devex LP code. *Mathematical Programming* **5** 1–28.

Kallio, M., E. L. Porteus. 1978. A class of methods for linear programming. *Mathematical Programming* **14** 161–169.

Maros, I. 2003. *Computational Techniques of the Simplex Method*. International Series in Operations Research and Management Science, Kluwer Academic Publishers.

McCormick, S. T., A. Shioura. 2000. Minimum ratio canceling is oracle polynomial for linear programming but not strongly polynomial even for networks. *Operations Research Letters* **27** 199–207.

Metrane, A., F. Soumis, I. Elhallaoui. 2010. Column generation decomposition with the degenerate constraints in the subproblem. *European Journal of Operational Research* **207** 37–44.

Murtagh, B.A., M.A. Saunders. 1978. Large-scale linearly constrained optimization. *Mathematical Programming* **14** 41–72.

Pan, P.-Q. 2008. A primal deficient-basis simplex algorithm for linear programming. *Applied Mathematics and Computation* **196** 898–912.

Perold, A. F. 1980. A degeneracy exploiting LU factorization for the simplex method. *Mathematical Programming* **19** 239–254.

Raymond, V., F. Soumis, A. Metrane, J. Desrosiers. 2010a. Positive edge: A pricing criterion for the identification of non-degenerate simplex pivots. Les Cahiers du GERAD G-2010-61, GERAD, Montreal, Quebec, Canada.

Raymond, V., F. Soumis, D. Orban. 2010b. A new version of the improved primal simplex for degenerate linear programs. *Computers & Operations Research* **37** 91–98.

Rosat, S., I. Elhallaoui, F. Soumis, A. Lodi. 2014. Integral simplex using decomposition with primal cuts. *Experimental Algorithms*, *Lecture Notes in Computer Science*, vol. 8504. 22–33.

Towhidi, M., J. Desrosiers, F. Soumis. 2014. The positive edge criterion within COIN-OR's CLP. *Computers & Operations Research* **49** 41 – 46.