EURO J Transp Logist (2016) 5:161–204
DOI 10.1007/s13676-015-0077-5

TUTORIAL

Tools for primal degenerate linear programs: IPS, DCA, and PE

Jean Bertrand Gauthier · Jacques Desrosiers · Marco E. Lübbecke

Received: 22 October 2013 / Accepted: 19 January 2015 / Published online: 27 March 2015

Abstract This paper describes three recent tools for dealing with primal degeneracy in linear programming. The first one is the improved primal simplex (IPS) algorithm which turns degeneracy into a possible advantage. The constraints of the original problem are dynamically partitioned based on the numerical values of the current basic variables. The idea is to work only with those constraints that correspond to nondegenerate basic variables. This leads to a row-reduced problem which decreases the size of the current working basis. The main feature of IPS is that it provides a nondegenerate pivot at every iteration of the solution process until optimality is reached. To achieve such a result, a negative reduced cost convex combination of the variables at their bounds is selected, if any. This pricing step provides a necessary and sufficient optimality condition for linear programming. The second tool is the dynamic constraint aggregation (DCA), a constructive strategy specifically designed for set partitioning constraints. It heuristically aims to achieve the properties provided by the IPS methodology. We bridge the similarities and differences of IPS and DCA on set partitioning models. The final tool is the positive edge (PE) rule. It capitalizes on the compatibility definition to determine the status of a column vector and the associated variable during the reduced cost computation. Within IPS, the selection of a compatible variable to enter the basis ensures a nondegenerate pivot, hence PE permits a trade-off between strict improvement and high, reduced cost degenerate pivots. This added value is obtained

J. B. Gauthier · J. Desrosiers (✉)
HEC Montréal and GERAD, 3000, chemin de la Côte-Sainte-Catherine, Montréal H3T 2A7, Canada
e-mail: jacques.desrosiers@hec.ca

J. B. Gauthier
e-mail: jean-bertrand.gauthier@hec.ca

M. E. Lübbecke
Operations Research, RWTH Aachen University, Kackertstraße 7, 52072 Aachen, Germany
e-mail: marco.luebbecke@rwth-aachen.de

without explicitly computing the updated column components in the simplex tableau. Ultimately, we establish tight bonds between these three tools by going back to the linear algebra framework from which emanates the so-called concept of subspace basis.

Keywords Primal simplex · Degeneracy · Combination of entering variables · Positive edge rule · Nondegenerate pivot algorithm · Dynamic Dantzig–Wolfe decomposition · Vector subspace

1 Introduction

When solving a linear program with the primal simplex (PS) algorithm (see Dantzig 1963), degeneracy comes in two flavors: degenerate solutions and degenerate pivots. The first case is a question of observation; it is a dichotomous state of the solution which either exhibits degenerate basic variables or does not. A basic solution is degenerate if at least one of its basic variables is degenerate, that is, at its lower or upper bound. Geometrically speaking, it corresponds to an over-represented vertex meaning that several equivalent bases are associated with the same solution. The second case is the algorithm's culprit in more ways than one. In fact, it is the only phenomenon which jeopardizes its convergence both theoretically and empirically. Degeneracy questions the efficiency of the PS algorithm and creates ambiguity in the post-analysis. On the one hand, degeneracy can affect the efficiency in obtaining an optimal solution because it creates redundant work. More specifically, a degenerate pivot amounts to trading one degenerate basic variable for a nonbasic one. Since no gain is made with respect to the objective function, it is in the aftermath of the computations that one ultimately realizes the wasted effort. It is even possible to cycle meaning that the PS algorithm moves through a series of bases eventually returning to an already visited one. If this happens indefinitely, PS may not even converge (Schrijver 1986). On the other hand, a by-product of the PS algorithm is the sensitivity analysis done after the optimization. Each constraint is associated with a dual variable whose value depends on the chosen basis. Since an optimal degenerate basis is not uniquely defined, it can mislead the interpretation of two otherwise equivalent solutions.

It should be noted that column generation, used to solve linear programs with a huge number of variables (Barnhart et al. 1998; Lübbecke and Desrosiers 2005), is a natural extension of PS, and as such suffers from degeneracy as well. With that being said, degeneracy is a phenomenon encountered particularly often for linear programming relaxations of combinatorial optimization problems. Set partitioning and set covering models are prominent examples of practical relevance: vehicle routing and crew scheduling problems (and many related problems in transportation and logistics) are most successfully formulated this way (Desrosiers et al. 1995; Desaulniers et al. 1998).

Degeneracy has been under scrutiny for practically as long as linear programming. We distinguish two lines of studies from the literature. The first aims to

eliminate degeneracy altogether and the other provides guidelines to alleviate its impact. On the first count, think of the work of (Charnes 1952) which revolves around modifying the polytope of the whole solution space in such a way that no two solutions ever share the same vertex. The concept amounts to right-hand side perturbations thus creating slight variations in the way the hyperplanes intersect. While the idea of eradicating degeneracy altogether is appealing, today's simplex codes use a more ad hoc strategy which sends us to the second count.

The contributions of Wolfe (1963) and Ryan and Osborne (1988) are abundant evidence that applying this strategy as necessary is highly effective. The perturbations are now applied in an adaptive manner and on a more local scale. *Stabilization* extends the idea of perturbation by incorporating dual information. Penalty functions, trust regions and expansion strategies are among the instrumental concepts of stabilization as described in the papers of du Merle et al. (1999) and Ben Amor et al. (2009). Column generation benefits from the latter as it tackles the particular sensitivity to the values of dual variables during the resolution process.

Numerous pivot rules have also been proposed to avoid performing degenerate pivots. In this regards, the work of Terlaky and Zhang (1993) is enlightening in many respects. Indeed, while many of these rules share common properties and sometimes even correspond to special cases of one another, they are distinguished according to certain properties: feasibility maintenance, anti-cycling feature and recursive nature. While there might have been hope about the performance of many of these rules, even nondegenerate instances can be difficult to optimize as supported by Klee and Minty (1972). The performance of a pivot rule may therefore be considered as a trade-off between its complexity and the savings it procures with respect to the number of iterations. The state of the art in terms of degenerate problems seems to be the Devex rule of Harris (1973), see Terlaky and Zhang (1993). We underline that regardless of their intricacies, all of these rules have a limited gain with respect to the absence of guaranteed efficiency. That is, zero step size pivots could still ensue from the chosen direction. The anti-cycling feature present in Bland (1977) or Fukuda (1982) ensures this behavior does not happen indefinitely. It is however generally accepted that taking expensive measures to protect against cycling is not worthwhile.

A new trend appears in the late 1990s with the paper of Pan (1998) who formulates a generic basis for degenerate solutions. Embedding this concept in a column generation scheme led to the dynamic constraint aggregation (DCA) algorithm of Elhallaoui et al. (2005, 2008) for the set partitioning problem. This problem lends itself particularly well to such a concept because of its peculiar structure. Indeed, it is this very structure that allows DCA to heuristically harness the power of a generic basis and quite often find strictly improving pivots. The paper of Elhallaoui et al. (2011) extends the algorithmic methodology with the improved primal simplex (IPS). As its name would have it, this extension takes place with regards to any linear programming problem. In a nut shell, the structure of a solution is preemptively taken into account in order to drive the next pivot in a strictly improving direction. That structure is dynamically updated with respect to the current solution.

The methodological paper at hand describes three tools for dealing with primal degeneracy. At the heart of this framework lies the search for so-called *compatible* column vectors and associated variables. Whether such a column exists as is in the original problem or is constructed as a convex combination of these, it corresponds to a direction in a polytope induced by a transformation of the original simplex. As such, we believe that IPS provides a better starting point from which the other two tools can benefit in terms of presentation. The second tool is of course DCA (which was incidentally designed prior to IPS) and the third is the positive edge (PE) rule (Raymond et al. 2010a; Towhidi et al. 2014). It is safe to say that DCA is a method steered by practical imperatives. Yet, explaining the reason behind its performance can now also be done in a straightforward manner in light of the IPS framework. PE is yet another example of benefits obtained from a higher level of abstraction. Indeed, while manipulating the set of compatible vectors can be computationally efficient, identifying said set can be time consuming for large problems. PE aims to simplify this verification by extracting the compatibility status during the reduced cost computation using the original column data.

The paper is organized as follows: Sect. 2 first exposes the theory of IPS with several hints to PS. By casting the linear algebra framework on our study, Sect. 3 presents another perspective of IPS. Section 4 addresses the more practical side with regard to several implementation choices. The importance of compatibility in the design of specialized applications is highlighted in Sect. 5. The similarities and differences between IPS and DCA are examined in Sect. 6, while Sect. 7 reveals PE. Various results from the literature are reported at the end of Sects. 4, 6 and 7 depending on the context of the underlying tool. Our conclusions end the paper in Sect. 8.

Motivation In the words of Perold (1980), *a great many degenerate iterations* is usually the resulting observation of degeneracy. As a matter of fact, it is not unusual to see that when an average of 20 % of basic columns are degenerate, 50 % of the iterations are degenerate. While the former statement gives a feel for the negative impact of degeneracy, the second statement rapidly frames it within a quantitative measure. The degenerate variables percentage of each basic solution encountered during the resolution process is averaged over the number of iterations. As such, it is certainly possible to characterize a linear program as degenerate if some basis exhibits such a quality, yet it is much more interesting to measure the extent of this pathology. The latter is based on empirical evidence. A linear program is thus said to have a *degeneracy level* of β %, where $\beta = 20$ corresponds to the average in Perold's example.

In the same vein, a whole class of linear programs can be qualified in the same manner by computing the mean of these values. For instance, assignment network problems have a degeneracy level of 50, or even 100 % if upper bounds are explicit. We do not know of any guidelines to state that family classes are degenerate, but it is fair to say that the level should be at least 20 %. In vehicle routing, it is immediate how degeneracy occurs: Constraints represent (often large numbers) of tasks to be covered by relatively few vehicles or crew members, that is, only few variables assume a positive value, especially in an integer solution.

Notation and terminology Vectors and matrices are written in bold face. We denote by $\mathbf{I}_\ell$ the $\ell \times \ell$ identity matrix and by $\mathbf{0}$ (resp. $\mathbf{1}$) a vector/matrix with all zeros (resp. ones) entries of contextually appropriate dimension. For a subset $I \subseteq \{1, \ldots, m\}$ of row indices and a subset $J \subseteq \{1, \ldots, n\}$ of column indices, we denote by $\mathbf{A}_{IJ}$ the sub-matrix of $\mathbf{A}$ containing the rows and columns indexed by I and J, respectively. We further use standard linear programming notation like $\mathbf{A}_J\mathbf{x}_J$, the subset of columns of $\mathbf{A}$ indexed by J multiplied by the corresponding sub-vector of variables $\mathbf{x}_J$. The lower case notation is reserved for vectors and uses the same subset index rules. In particular, the matrix $\mathbf{A} := (\mathbf{a}_j)_{j\in\{1,\ldots,n\}}$ contains n column vectors. Finally, there is one notable exception: The set N does *not* denote the nonbasis but rather the set of basic and nonbasic variables at their lower or upper bounds. Hence, for a linear program in standard form, $\mathbf{x}_N$ represents the vector of null variables.

The pricing step in the seminal IPS papers refers to solving a *complementary problem* whereas it was later shown that IPS can be seen as a dynamic Dantzig–Wolfe decomposition at every iteration. As a survey paper, we use a unifying terminology and choose to define the pricing step as solving a *pricing problem*.

2 Improved primal simplex

This section first exposes the theory of IPS in the context of a linear program with lower and upper bounded variables. It is based on the original papers of Elhallaoui et al. (2011), Raymond et al. (2009, 2010b), Metrane et al. (2010) and its generalization to row-reduced column generation (Desrosiers et al. 2014). However, contrary to the original presentation, the choice of using a bounded linear program in the description of IPS is becoming of its purpose. For instance, in set partitioning problems, degenerate upper bounds are exploited for a faster resolution. Moreover, the change of variables utilized for the row partition also becomes more apparent with upper bounds.

We present in Sect. 2.1 the algorithmic steps of IPS. Section 2.2 provides the proof of a necessary and sufficient optimality condition derived from the improved pricing step. Section 2.3 presents a simplified version of IPS for linear programs in standard form. For a better understanding of the concepts, an illustrative example is given in Sect. 2.4 on a small linear program.

Consider a linear program (LP) with lower and upper bounded variables:

$$\begin{array}{rlll} z^\star := & \min & \mathbf{c}^\intercal\mathbf{x} & \\ & \text{s.t.} & \mathbf{A}\mathbf{x} & = \mathbf{b}, \quad [\boldsymbol{\pi}] \\ & & \mathbf{l} \le \mathbf{x} \le \mathbf{u}, & \end{array} \tag{1}$$

where $\mathbf{x}, \mathbf{c}, \mathbf{l}, \mathbf{u} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{A} \in \mathbb{R}^{m\times n}$, and $m < n$. We assume that $\mathbf{A}$ is a matrix of full row rank and that LP is feasible and bounded. Finally, $\boldsymbol{\pi} \in \mathbb{R}^m$ is a vector of dual variables associated with the equality constraints.

2.1 Algorithmic steps

The main idea in IPS is to reduce the number of constraints from m to f, the number of nondegenerate or free variables in a basic solution. The advantage of this row reduction is a smaller working basis of dimension $f \times f$ rather than the usual larger one of dimension $m \times m$. This comes at the expense of a more involved pricing step which solves a linear program of row size $m - f + 1$ to select an improving subset of columns, that is, a convex combination of columns with two properties: this selection is compatible with the current row-reduced problem (see Definition 1) and its reduced cost is negative. If such a combination exists, a strict improvement in the objective function value occurs, otherwise the current solution is optimal. Figure 1 contains an overview of the main steps of IPS. The initialization contains the change of variables, the input basic solution $\mathbf{x}^0$, and the associated column partition with null variables set N. The main loop provides: (1) the construction of a generic basis and the resulting linear transformation and row partition; (2) the definition of compatibility; (3) the development of an improving pricing step; (4) the exchange mechanism from a solution $\mathbf{x}^0$ to the next $\mathbf{x}^1$ which incidentally brings an inspiring twist to the pivoting rule; (5) the update of the column partition.

Initialization Let $\mathbf{x}^0$, represented by $(\mathbf{x}^0_F; \mathbf{x}^0_L; \mathbf{x}^0_U)$, be a basic solution where the three sub-vectors are defined according to the value of their variables: $\mathbf{x}^0_L$ at their lower bounds, $\mathbf{x}^0_U$ at their upper bounds, and *free* variables $\mathbf{l}_F < \mathbf{x}^0_F < \mathbf{u}_F$. Free variables are basic and they can move below or above their current value which obviously makes them nondegenerate. Let there be $f := |F|$ such free variables, $0 \leq f \leq m$. Partition the matrix $\mathbf{A} = [\mathbf{A}_F, \mathbf{A}_L, \mathbf{A}_U]$ and cost vector $\mathbf{c}^\intercal = [\mathbf{c}^\intercal_F, \mathbf{c}^\intercal_L, \mathbf{c}^\intercal_U]$ accordingly. Although the change of variables is blindly applied, IPS retains only those pertinent to the construction:

$$\begin{aligned} \mathbf{y}_L &:= \mathbf{x}_L - \mathbf{x}^0_L, \quad \mathbf{y}_L \geq \mathbf{0} \\ \mathbf{y}_U &:= \mathbf{x}^0_U - \mathbf{x}_U, \quad \mathbf{y}_U \geq \mathbf{0}. \end{aligned} \tag{2}$$

Let $N := L \cup U$ to form $\mathbf{y}_N = (\mathbf{y}_L; \mathbf{y}_U)$, the vector of currently null $\mathbf{y}$-variables, bounded above by $\mathbf{r}_N$, where $r_j := u_j - \ell_j, \forall j \in N$. Let $\mathbf{d}^\intercal_N := [\mathbf{c}^\intercal_L, -\mathbf{c}^\intercal_U]$ and define $\mathbf{A}^0_N := [\mathbf{A}_L, -\mathbf{A}_U]$, that is, $\mathbf{a}^0_j = \mathbf{a}_j$, $\forall j \in L$, and $\mathbf{a}^0_j = -\mathbf{a}_j$, $\forall j \in U$. Given the adjusted right-hand side $\mathbf{b}^0 := \mathbf{b} - \mathbf{A}_L\mathbf{x}^0_L - \mathbf{A}_U\mathbf{x}^0_U$, LP becomes:

Initialization: basic solution $\mathbf{x}^0$;
change of variables;
column partition $\{F, L, U\}$ and $N := \{L \cup U\}$;
1 Generic basis $\mathbf{B}$, transformation $\mathbf{B}^{-1}$, row partition $\{P, Z\}$ of $\mathbf{A}_F$;
2 Compatibility with the row partition $\{P, Z\}$ of $\mathbf{A}_F$ <optional>;
3 Improved pricing step: optimize the minimum reduced cost μ;
4 Exchange mechanism from $\mathbf{x}^0$ to $\mathbf{x}^1$;
5 Update the column partition $\{F, L, U\}$ and goto Step 1;

Fig. 1 IPS algorithmic steps

$$z^\star = \mathbf{c}_L^\intercal \mathbf{x}_L^0 + \mathbf{c}_U^\intercal \mathbf{x}_U^0 + \begin{array}{lccc} \min & \mathbf{c}_F^\intercal \mathbf{x}_F & + & \mathbf{d}_N^\intercal \mathbf{y}_N & \\ \text{s.t.} & \mathbf{A}_F \mathbf{x}_F & + & \mathbf{A}_N^0 \mathbf{y}_N & = \mathbf{b}^0, \quad [\boldsymbol{\pi}] \\ & \mathbf{l}_F \le \mathbf{x}_F \le \mathbf{u}_F, & & \mathbf{0} \le \mathbf{y}_N \le \mathbf{r}_N. & \end{array} \tag{3}$$

1. *Generic basis* $\mathbf{B}$, *transformation* $\mathbf{B}^{-1}$, *row partition* $\{P, Z\}$ *of* $\mathbf{A}_F$ The current solution being basic, the columns of $\mathbf{A}_F$ are linearly independent. When $f = m$, there is no row reduction but the current solution is nondegenerate, and so is the next pivot. Assume that $f < m$ such that the basis associated with $\mathbf{x}^0$ contains degenerate variables. Let us call *basis completion* the process of selecting $m - f$ variables taking value zero which complement $\mathbf{A}_F$ by forming a nonsingular basis matrix. Since any and all combinations of degenerate variables which may complete the basis is as good as the next one, let us construct a *generic* $m \times m$ basis denoted $\mathbf{B}$. Such a basis is readily available using the f free variables associated with the columns of $\mathbf{A}_F$ together with $m - f$ artificial variables. The selection of an appropriate set of artificial variables can be done by solving a *restricted* primal simplex phase I problem over the columns of $\mathbf{A}_F$ and those of the identity matrix $\mathbf{I}_m$ with the corresponding vector of artificial variables here denoted $\boldsymbol{\lambda}$:

$$\begin{array}{lccc} \min & & \mathbf{1}^\intercal \boldsymbol{\lambda} & \\ \text{s.t.} & \mathbf{A}_F \mathbf{x}_F & + \quad \mathbf{I}_m \boldsymbol{\lambda} & = \mathbf{b}^0, \\ & \mathbf{x}_F \ge \mathbf{0}, & \boldsymbol{\lambda} \ge \mathbf{0}. & \end{array} \tag{4}$$

Solving this problem is undoubtedly successful in accordance with the fact that $\mathbf{A}_F \mathbf{x}_F^0 = \mathbf{b}^0$. Furthermore, this restricted phase I differs from a cold start phase I on one key point: only the former can guarantee a basis in which all degenerate basic variables are artificial ones. Let it be clear that this construction process identifies some subset $\mathbf{A}_{PF}$ of exactly f independent rows from matrix $\mathbf{A}_F$. This provides the row partition $\{P, \bar{P}\}$ of $\mathbf{A}_F$, where we use $Z := \bar{P}$ for notational convenience. The generic basis $\mathbf{B}$ and its inverse $\mathbf{B}^{-1}$ are as follows:

$$\mathbf{B} = \begin{bmatrix} \mathbf{A}_{PF} & \mathbf{0} \\ \mathbf{A}_{ZF} & \mathbf{I}_{m-f} \end{bmatrix} \quad \text{and} \quad \mathbf{B}^{-1} = \begin{bmatrix} \mathbf{A}_{PF}^{-1} & \mathbf{0} \\ -\mathbf{A}_{ZF}\mathbf{A}_{PF}^{-1} & \mathbf{I}_{m-f} \end{bmatrix}, \tag{5}$$

where the matrix $\mathbf{A}_{PF}$ of dimension $f \times f$ is the working basis. The basis $\mathbf{B}$ is one of the many bases available to identify the over-represented vertex $\mathbf{x}^0$. As such, observe the sensitivity of the dual vector $\boldsymbol{\pi}^\intercal := \mathbf{c}_B^\intercal \mathbf{B}^{-1}$ with respect to the choice of basis completion. LP becomes

$$z^\star = \mathbf{c}_L^\intercal \mathbf{x}_L^0 + \mathbf{c}_U^\intercal \mathbf{x}_U^0 + \begin{array}{lccc} \min & \mathbf{c}_F^\intercal \mathbf{x}_F & + & \mathbf{d}_N^\intercal \mathbf{y}_N & \\ \text{s.t.} & \mathbf{A}_{PF} \mathbf{x}_F & + & \mathbf{A}_{PN}^0 \mathbf{y}_N & = \mathbf{b}_P^0, \quad [\boldsymbol{\pi}_P] \\ & \mathbf{A}_{ZF} \mathbf{x}_F & + & \mathbf{A}_{ZN}^0 \mathbf{y}_N & = \mathbf{b}_Z^0, \quad [\boldsymbol{\pi}_Z] \\ & \mathbf{l}_F \le \mathbf{x}_F \le \mathbf{u}_F, & & \mathbf{0} \le \mathbf{y}_N \le \mathbf{r}_N. & \end{array} \tag{6}$$

Let $\bar{\mathbf{b}}^0 := \mathbf{B}^{-1}\mathbf{b}^0$ and

$$\bar{\mathbf{A}}_N^0 := \mathbf{B}^{-1}\mathbf{A}_N^0 = \begin{bmatrix} \bar{\mathbf{A}}_{PN}^0 \\ \bar{\mathbf{A}}_{ZN}^0 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{PF}^{-1}\mathbf{A}_{PN}^0 \\ \mathbf{A}_{ZN}^0 - \mathbf{A}_{ZF}\mathbf{A}_{PF}^{-1}\mathbf{A}_{PN}^0 \end{bmatrix}. \tag{7}$$

The new LP formulation obtained after the change of $\mathbf{y}$-variables and the left-multiplication by the linear transformation $\mathbf{B}^{-1}$ of the set of equality constraints (which incidentally also transforms the dual variables) makes degeneracy more evident:

$$\begin{array}{rlccll} z^\star = \mathbf{c}_L^\intercal \mathbf{x}_L^0 + \mathbf{c}_U^\intercal \mathbf{x}_U^0 + \min & \mathbf{c}_F^\intercal \mathbf{x}_F & + & \mathbf{d}_N^\intercal \mathbf{y}_N & & \\ \text{s.t.} & \mathbf{x}_F & + & \bar{\mathbf{A}}_{PN}^0 \mathbf{y}_N & = \bar{\mathbf{b}}_P^0, & [\boldsymbol{\psi}_P] \\ & & & \bar{\mathbf{A}}_{ZN}^0 \mathbf{y}_N & = \mathbf{0}, & [\boldsymbol{\psi}_Z] \\ & \mathbf{l}_F \le \mathbf{x}_F \le \mathbf{u}_F, & & \mathbf{0} \le \mathbf{y}_N \le \mathbf{r}_N. & & \end{array} \tag{8}$$

The current solution is given by $\mathbf{x}_F = \mathbf{x}_F^0 = \mathbf{A}_{PF}^{-1}\mathbf{b}_P^0 = \bar{\mathbf{b}}_P^0$ while $\mathbf{y}_N = \mathbf{0}$. Observe that the constraints of LP are divided according to the actual values of $\bar{\mathbf{b}}^0$: for the row set P, $\bar{\mathbf{b}}_P^0 > \mathbf{0}$; for the remaining rows in the set Z, $\bar{\mathbf{b}}_Z^0 = \mathbf{0}$. The dual vector $\boldsymbol{\pi}$ can be retrieved from the above transformed dual vector $\boldsymbol{\psi}$ using the expression $\boldsymbol{\pi}^\intercal = \boldsymbol{\psi}^\intercal \mathbf{B}^{-1}$:

$$\boldsymbol{\pi}_P^\intercal = \boldsymbol{\psi}_P^\intercal \mathbf{A}_{PF}^{-1} - \boldsymbol{\psi}_Z^\intercal \mathbf{A}_{ZF}\mathbf{A}_{PF}^{-1} \tag{9}$$

$$\boldsymbol{\pi}_Z^\intercal = \boldsymbol{\psi}_Z^\intercal. \tag{10}$$

2. *Compatibility with the row partition* $\{P, Z\}$ *of* $\mathbf{A}_F$ Observe that any solution to (8), optimal or not, must satisfy $\bar{\mathbf{A}}_{ZN}^0 \mathbf{y}_N = \mathbf{0}$. This leads us to the first definition of compatibility.

Definition 1 A vector $\mathbf{a} \in \mathbb{R}^m$ (and the associated variable, if any) is *compatible with the row partition* $\{P, Z\}$ *of* $\mathbf{A}_F$ if and only if $\bar{\mathbf{a}}_Z := \mathbf{a}_Z - \mathbf{A}_{ZF}\mathbf{A}_{PF}^{-1}\mathbf{a}_P = \mathbf{0}$.

One can derive from the formulation (8) that the column vectors of $\mathbf{A}_F$ are compatible (hence the free variables x_j, $j \in F$) as well as the transformed right-hand side vector $\mathbf{b}^0$ (with no associated variable) but degenerate basic variables are not.

3. *Improved pricing step: optimize minimum reduced cost* μ. The variables $\mathbf{x}_F$ are basic in the row set P, hence the reduced cost vector $\bar{\mathbf{c}}_F = \mathbf{c}_F - \boldsymbol{\psi}_P = \mathbf{0}$ which means that $\boldsymbol{\psi}_P = \mathbf{c}_F$. With respect to the values of $\boldsymbol{\psi}_Z$, we recall the basis completion paradigm whereby the selection of degenerate variables that complete the basis influences the values of their associated dual variables. In other words, it is possible to capitalize on this freedom and consider them undetermined. The current solution $\mathbf{x}^0 = (\mathbf{x}_F^0; \mathbf{x}_L^0; \mathbf{x}_U^0)$ is optimal for (1), or equivalently $(\mathbf{x}_F; \mathbf{y}_N) = (\mathbf{x}_F^0; \mathbf{0})$ is optimal for (8), if there exists some dual vector $\boldsymbol{\psi}_Z$ such that the reduced cost $\bar{d}_j$ of every variable y_j, $j \in N$, is nonnegative, that is, $\bar{d}_j := d_j - \mathbf{c}_F^\intercal \bar{\mathbf{a}}_{Pj}^0 - \boldsymbol{\psi}_Z^\intercal \bar{\mathbf{a}}_{Zj}^0 \ge 0$, $\forall j \in N$.

Let $\mu := \min_{j \in N} \bar{d}_j$ be the smallest reduced cost for $\mathbf{y}_N$ given $\boldsymbol{\psi}_P = \mathbf{c}_F$ but optimized over $\boldsymbol{\psi}_Z$. Finding μ can be formulated as a linear program:

$$\begin{array}{lll} \max & \mu & \\ \text{s.t.} & \mu & \leq \quad d_j - \mathbf{c}_F^\intercal \bar{\mathbf{a}}_{Pj}^0 - \boldsymbol{\psi}_Z^\intercal \bar{\mathbf{a}}_{Zj}^0, \quad [y_j] \quad \forall j \in N, \end{array} \tag{11}$$

where $y_j \geq 0$, $j \in N$, is the dual variable associated with the corresponding inequality constraint. Let $\tilde{d}_j := d_j - \mathbf{c}_F^\intercal \bar{\mathbf{a}}_{Pj}^0$ be the *partial reduced cost* of y_j computed by using dual vector $\boldsymbol{\psi}_P = \mathbf{c}_F$, or equivalently $\tilde{\mathbf{d}}_N^\intercal := \mathbf{d}_N^\intercal - \mathbf{c}_F^\intercal \bar{\mathbf{A}}_{PN}^0$ in vector form. Therefore, (11) becomes

$$\begin{array}{lll} \max & \mu & \\ \text{s.t.} & \mathbf{1}\mu \quad + \boldsymbol{\psi}_Z^\intercal \bar{\mathbf{A}}_{ZN}^0 \leq \tilde{\mathbf{d}}_N, & [\mathbf{y}_N]. \end{array} \tag{12}$$

Taking the dual of (12), the pricing problem is written in terms of $\mathbf{y}_N$, the vector of currently null variables to price out:

$$\begin{array}{lllll} \mu = & \min & \tilde{\mathbf{d}}_N^\intercal \mathbf{y}_N & & \\ & \text{s.t.} & \mathbf{1}^\intercal \mathbf{y}_N & = 1, & [\mu] \\ & & \bar{\mathbf{A}}_{ZN}^0 \mathbf{y}_N & = \mathbf{0}, & [\boldsymbol{\psi}_Z] \\ & & \mathbf{y}_N & \geq \mathbf{0}. & \end{array} \tag{13}$$

The pricing problem (13) can be solved by the dual simplex algorithm because only the convexity constraint $\mathbf{1}^\intercal \mathbf{y}_N = 1$ is not satisfied by the current value $\mathbf{y}_N = \mathbf{0}$. For a more recent analysis of the resolution of the pricing problem, Omer et al. (2014) explore ways to warm start the basis notably with the use of more elaborate coefficients for the convexity constraints. Alternatively, specialized algorithms can be used in some applications. This is the case for LP defined as a capacitated minimum cost network flow problem where the pricing problem (13) corresponds to a minimum mean cost cycle problem which can be solved in $O(mn)$ time by dynamic programming Karp (1978). What ultimately matters is that we are looking for extreme point solutions to (13) (see Gauthier et al. (2014)).

The number of positive variables in an optimal solution $\mathbf{y}_N^0$ to (13) is at most $m - f + 1$, the row dimension of the pricing problem. The solution $\mathbf{x}^0$ is optimal for LP if $\mu \geq 0$. Otherwise, $\mu < 0$ and $\mathbf{y}_N^0$ identifies a convex combination of columns such that $\bar{\mathbf{A}}_{ZN}^0 \mathbf{y}_N^0 = \mathbf{0}$. Observe that by Definition 1, the vector $\mathbf{A}_N^0 \mathbf{y}_N^0 \in \mathbb{R}^m$ is compatible with the row partition $\{P, Z\}$ of $\mathbf{A}_F$. Let $\boldsymbol{\Omega}$ be the set of all such compatible convex combinations of columns.

4. *Exchange mechanism from* $\mathbf{x}^0$ *to* $\mathbf{x}^1$. The solution $\mathbf{y}_N^0$ is utilized to move from $\mathbf{x}^0$ to $\mathbf{x}^1$. Let the compatible column $\mathbf{A}_N^0 \mathbf{y}_N^0$ be associated with a surrogate variable θ_ω, $\omega \in \boldsymbol{\Omega}$, nonexistent from the original formulation. Parameters of θ_ω relative to formulation (8) are as follows: $\bar{\mathbf{a}}_w^0 = \begin{bmatrix} \bar{\mathbf{a}}_{Pw}^0 \\ \bar{\mathbf{a}}_{Zw}^0 \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{A}}_{PN}^0 \mathbf{y}_N^0 \\ \mathbf{0} \end{bmatrix}$, reduced cost μ, cost $\mathbf{d}_N^\intercal \mathbf{y}_N^0$, and $\mathbf{y}_N^0 \neq \mathbf{0}$. With the addition of the variable θ_ω to the LP model in (8), we have the following relations, where relevant parameters are indicated within brackets:

$$\begin{array}{lll} \mathbf{x}_F & + & [\bar{\mathbf{a}}^0_{P\omega}]\,\theta_\omega & = \bar{\mathbf{b}}^0_P, \\ \mathbf{l}_F \le \mathbf{x}_F \le \mathbf{u}_F, & & 0 \le [\mathbf{y}^0_N]\;\theta_\omega \le \mathbf{r}_N. \end{array} \tag{14}$$

Regardless of its solution, observe that the pricing problem finds a *partial* improving direction $\mathbf{y}^0_N$ of negative reduced cost value μ, if one exists, uniquely completed by $\mathbf{y}^0_F$, the impact in the row set P:

$$\begin{bmatrix} \mathbf{y}^0_F \\ \mathbf{y}^0_N \end{bmatrix} = \begin{bmatrix} -\bar{\mathbf{a}}^0_{P\omega} \\ \mathbf{y}^0_N \end{bmatrix} = \begin{bmatrix} -\bar{\mathbf{A}}^0_{PN}\mathbf{y}^0_N \\ \mathbf{y}^0_N \end{bmatrix} \in \mathbb{R}^n. \tag{15}$$

The step size ρ is governed by the usual pivot rule. In (14), the entering variable θ_ω can increase up to the maximum change for $\mathbf{y}_N$, that is, $\mathbf{y}^0_N\theta_\omega \le \mathbf{r}_N$, or according to the maximum change for $\mathbf{x}_F$, that is, $\mathbf{l}_F \le \bar{\mathbf{b}}^0_P - \bar{\mathbf{a}}^0_{P\omega}\theta_\omega \le \mathbf{u}_F$. The step size ρ on θ_ω is given by

$$\rho := \min\left\{ \min_{j\in N|y^0_j>0}\left\{\frac{r_j}{y^0_j}\right\}, \min_{i\in P|\bar{a}^0_{i\omega}>0}\left\{\frac{\bar{b}^0_i - l_i}{\bar{a}^0_{i\omega}}\right\}, \min_{i\in P|\bar{a}^0_{i\omega}<0}\left\{\frac{u_i - \bar{b}^0_i}{-\bar{a}^0_{i\omega}}\right\}\right\}. \tag{16}$$

A nondegenerate pivot occurs ($\rho > 0$) and the objective function strictly improves by

$$\Delta z = \rho\mu = \rho\,\tilde{\mathbf{d}}^{\mathsf{T}}_N\mathbf{y}^0_N. \tag{17}$$

The solution $\mathbf{x}^0$ is updated to $\mathbf{x}^1$ according to the direction expression in (15):

$$\begin{aligned} \mathbf{x}^1_F &:= \mathbf{x}^0_F - \rho\,\bar{\mathbf{a}}^0_{P\omega} \\ \mathbf{x}^1_L &:= \mathbf{x}^0_L + \rho\,\mathbf{y}^0_L \\ \mathbf{x}^1_U &:= \mathbf{x}^0_U - \rho\,\mathbf{y}^0_U. \end{aligned} \tag{18}$$

The number of free variables in $\mathbf{x}^1$ is at most $f + (m - f + 1) - 1 = m$, that is, the new solution can be more degenerate but it can also be less degenerate when several variables of the convex combination become free.

Regardless of the manner in which one updates the current solution, the aftermath is the result of an *exchange mechanism*. Even the ratio test performed to identify the exiting variable in PS echoes this notion. Indeed, the exchange always happens in a one-to-one fashion, while we have just seen that it can be more involved. Given the current solution, the exchange mechanism provided in (18) starts in the pricing problem (13) for the rows in set Z by finding in $\bar{\mathbf{A}}^0_{ZN}\mathbf{y}_N = \mathbf{0}, \mathbf{y}_N \ge \mathbf{0}$, which induces the partial directions $\mathbf{y}^0_L$ and $-\mathbf{y}^0_U$ for the vectors $\mathbf{x}_L$ and $\mathbf{x}_U$, respectively. The exchange process is afterward completed by using the rows in set P and interval constraints in (14): the partial direction for the vector $\mathbf{x}_F$ is given by $-\bar{\mathbf{a}}^0_{P\omega} = -\bar{\mathbf{A}}^0_{PN}\mathbf{y}^0_N$ and the step size is derived in expression (16). In the latter, it occurs between $\mathbf{x}_F$ and the entering variable $\theta_\omega, \omega \in \mathbf{\Omega}$.

5. *Update column partition* $\{F, L, U\}$ In the midst of obtaining the new solution $\mathbf{x}^1$, every variable affected by the direction is identified. It is therefore easy

to modify the status of each of these variables if necessary. Notice that the generic basis $\mathbf{B}$ is inspired by the column partition F.

Special case $y_j^0 = 1,\ j \in N$. The reader is invited to contemplate the special case where the convex combination contains a single variable $y_j^0,\ j \in N$. The repercussions are many in terms of mathematical simplifications but we are most interested in the following one. The surrogate variable actually exists as is in the original formulation (8) which means that some existing variables in set N are compatible with the row partition $\{P, Z\}$ of $\mathbf{A}_F$. In that case, the column $\mathbf{a}_j^0,\ j \in N$, enters the basis $\mathbf{B}$, $\mu = \tilde{d}_j$, and the step size ρ is computed according to the maximum increase of variable y_j. From (8), we have the following relations:

$$\begin{array}{ccc} \mathbf{x}_F & + & \bar{\mathbf{a}}_{Pj}^0 y_j = \bar{\mathbf{b}}_P^0, \\ \mathbf{l}_F \le \mathbf{x}_F \le \mathbf{u}_F, & & 0 \le y_j \le r_j. \end{array} \tag{19}$$

The step size ρ on y_j can increase up to the upper bound r_j, or according to the maximum change in the vector of free variables $\mathbf{l}_F \le \bar{\mathbf{b}}_P^0 - \bar{\mathbf{a}}_{Pj}^0 y_j \le \mathbf{u}_F$:

$$\rho := \min\left\{ r_j,\ \min_{i \in P | \bar{a}_{ij}^0 > 0} \left\{ \frac{\bar{b}_i^0 - l_i}{\bar{a}_{ij}^0} \right\},\ \min_{i \in P | \bar{a}_{ij}^0 < 0} \left\{ \frac{u_i - \bar{b}_i^0}{-\bar{a}_{ij}^0} \right\} \right\} > 0\,. \tag{20}$$

The objective function z improves by $\Delta z = \rho \tilde{d}_j = \rho\mu$. Either $j \in L$ (x_j is at its lower bound) or $j \in U$ (x_j is at its upper bound) and $\mathbf{x}^0$ is updated to $\mathbf{x}^1$ as

$$\begin{array}{rcl} \mathbf{x}_F^1 & := & \mathbf{x}_F^0 - \rho\, \bar{\mathbf{a}}_{Pj}^0 \\ \mathbf{x}_L^1 & := & \mathbf{x}_L^0 + \rho\, \mathbf{y}_L^0 \\ \mathbf{x}_U^1 & := & \mathbf{x}_U^0 - \rho\, \mathbf{y}_U^0. \end{array} \tag{21}$$

The number of free variables in $\mathbf{x}^1$ is at most f, that is, the new solution can be more degenerate. If $\rho < r_j$, f decreases if more than one of the free variables reach their bounds. Otherwise $\rho = r_j$, the corresponding x_j variable changes bound and therefore stays degenerate in the new solution; the number of free variables decreases if at least one free variable reaches a bound.

2.2 Characterization of linear programming optimality

In summary, when $\mu \ge 0$, the current solution $\mathbf{x}^0$ is optimal. Otherwise, $\mu < 0$ and we obtain a strict improvement of the objective function, update the current solution from $\mathbf{x}^0$ to $\mathbf{x}^1$, and the process is repeated until the following *necessary and sufficient optimality condition* is met.

Proposition 1 *A basic feasible solution* $\mathbf{x}^0 = (\mathbf{x}_F^0; \mathbf{x}_L^0; \mathbf{x}_U^0)$ *is an optimal solution to the linear program* (1) *if and only if there exists a dual vector* $\boldsymbol{\psi}_Z$ *such that* $\mu \ge 0$ *, as optimized by the primal-dual pair* (12)–(13) *of the pricing problem.*

Proof The formulations (1) and (8) are equivalent. Because $\bar{\mathbf{c}}_F = \mathbf{0}$, if there exists some dual vector $\boldsymbol{\psi}_Z$ such that $\mathbf{d}_N^\intercal - \mathbf{c}_F^\intercal \bar{\mathbf{A}}_{PN}^0 - \boldsymbol{\psi}_Z^\intercal \bar{\mathbf{A}}_{ZN}^0 \ge \mathbf{0}^\intercal$, $N := L \cup U$, then

$(\bar{\mathbf{c}}_F, \bar{\mathbf{d}}_N) \geq \mathbf{0}$. Therefore, $\boldsymbol{\psi}^\intercal = (\mathbf{c}_F^\intercal, \boldsymbol{\psi}_Z^\intercal)$ provides a feasible dual solution to (8). Since $\boldsymbol{\psi}_P^\intercal \bar{\mathbf{b}}_P^0 = \mathbf{c}_F^\intercal \mathbf{x}_F^0$, the primal and dual objective functions are equal and the current feasible solution $\mathbf{x}^0$ is optimal for (1).

To show the converse, let $\mathbf{x}^0$ be an optimal solution to (1) and assume that $\mu < 0$. An optimal solution to the pricing problem (13) identifies a convex combination of variables such that a nondegenerate pivot occurs ($\rho > 0$) and the objective function strictly improves by $\rho\mu < 0$. This contradicts the optimality of $\mathbf{x}^0$ and completes the proof. □

All simplex derivatives work according to the presumption of innocence. Optimality is indeed assumed until proven otherwise. It is no different in IPS, yet it is an amazing feat that the content of the pricing problem be reminiscent of the no more, no less punch line. The sufficient condition answers to the first part, while the necessary condition to the second.

2.3 IPS for a linear program in standard form

The reader may recall that incorporating lower and upper bounds in PS adds a plethora of intricacies in the algorithmic analysis. Although the same is true of IPS, we assumed the reader was sufficiently accustomed with the traditional algorithm. In the spirit of conveying the general idea of IPS, it might be worthwhile to present a simpler version. This basically amounts to removing the dimension U from the formulation. The simplifications are threesome and correspond to the main steps of IPS: creating the column and row partitions, building the pricing problem, and modifying the current solution. Given LP in standard form

$$\begin{array}{rlll} z^\star := & \min & \mathbf{c}^\intercal \mathbf{x} & \\ & \text{s.t.} & \mathbf{A}\mathbf{x} = \mathbf{b}, & [\boldsymbol{\pi}] \\ & & \mathbf{x} \geq \mathbf{0}, & \end{array} \tag{22}$$

and a feasible solution $\mathbf{x}^0 = (\mathbf{x}_F^0; \mathbf{x}_N^0)$, the column partition step distinguishes between the currently nondegenerate (or free) basic vector $\mathbf{x}_F^0$ and null vector $\mathbf{x}_N^0$:

$$\begin{array}{rlllll} z^\star = & \min & \mathbf{c}_F^\intercal \mathbf{x}_F & + & \mathbf{c}_N^\intercal \mathbf{x}_N & \\ & \text{s.t.} & \mathbf{A}_F \mathbf{x}_F & + & \mathbf{A}_N \mathbf{x}_N = \mathbf{b}, & [\boldsymbol{\pi}] \\ & & \mathbf{x}_F \geq \mathbf{0}, & & \mathbf{x}_N \geq \mathbf{0}. & \end{array} \tag{23}$$

Recall the previous change of variables in (2). Since N now only contains variables at their lower bounds, $\mathbf{x}_N$ could be used interchangeably with $\mathbf{y}_N$. We maintain the general presentation to underscore that the construction aims to find an improving direction induced by $\mathbf{y}_N$. It should also be clear that $\mathbf{A}_N^0 = \mathbf{A}_N$, and $\mathbf{b}^0 = \mathbf{b}$.

$$
\begin{array}{rlllll}
z^{\star} = & \min & \mathbf{c}_F^{\mathsf{T}}\mathbf{x}_F & + & \mathbf{c}_N^{\mathsf{T}}\mathbf{y}_N & \\
& \text{s.t.} & \mathbf{x}_F & + & \bar{\mathbf{A}}_{PN}\mathbf{y}_N = \bar{\mathbf{b}}_P, & [\boldsymbol{\psi}_P] \\
& & & & \bar{\mathbf{A}}_{ZN}\mathbf{y}_N = \mathbf{0}, & [\boldsymbol{\psi}_Z] \\
& & \mathbf{x}_F \geq \mathbf{0}, & & \mathbf{y}_N \geq \mathbf{0}. &
\end{array}
\tag{24}
$$

Once again, the linear transformation $\mathbf{B}^{-1}$ performed on the original system underlines the degeneracy of the current solution. Furthermore, any solution must satisfy $\bar{\mathbf{A}}_{ZN}\mathbf{y}_N = \mathbf{0}$ in (24). Therefore, the pricing problem can be written in terms of the vector of null variables to price out, and the current partial reduced cost vector $\tilde{\mathbf{c}}_N^{\mathsf{T}} := \mathbf{c}_N^{\mathsf{T}} - \boldsymbol{\psi}_P^{\mathsf{T}}\bar{\mathbf{A}}_{PN} = \mathbf{c}_N^{\mathsf{T}} - \mathbf{c}_F^{\mathsf{T}}\bar{\mathbf{A}}_{PN}$:

$$
\begin{array}{rlll}
\mu := & \min & \tilde{\mathbf{c}}_N^{\mathsf{T}}\mathbf{y}_N & \\
& \text{s.t.} & \mathbf{1}^{\mathsf{T}}\mathbf{y}_N = 1, & [\mu] \\
& & \bar{\mathbf{A}}_{ZN}\mathbf{y}_N = \mathbf{0}, & [\boldsymbol{\psi}_Z] \\
& & \mathbf{y}_N \geq \mathbf{0}. &
\end{array}
\tag{25}
$$

The solution $\mathbf{x}^0 = (\mathbf{x}_F^0 > \mathbf{0}; \mathbf{x}_N^0 = \mathbf{0})$ is optimal for LP in (22) if $\mu \geq 0$. Otherwise $\mu < 0$ and an optimal solution $\mathbf{y}_N^0$ to (25) identifies a convex combination of variables such that $\bar{\mathbf{A}}_{ZN}\mathbf{y}_N^0 = \mathbf{0}$. The convex combination established by the pricing problem may once again contain one or several $\mathbf{y}$-variables. Let $\theta_\omega, \omega \in \boldsymbol{\Omega}$, be the entering variable with the following parameters: reduced cost μ, cost $\mathbf{c}_N^{\mathsf{T}}\mathbf{y}_N^0$, and $\bar{\mathbf{a}}_\omega = \begin{bmatrix} \bar{\mathbf{a}}_{P\omega} \\ \bar{\mathbf{a}}_{Z\omega} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{A}}_{PN}\mathbf{y}_N^0 \\ \mathbf{0} \end{bmatrix}$. What matters is that the ratio test (16) is now computed with a single component:

$$
\rho := \min_{i \in P | \bar{a}_{i\omega} > 0} \left\{ \frac{\bar{b}_i}{\bar{a}_{i\omega}} \right\} > 0. \tag{26}
$$

A nondegenerate pivot occurs and LP's objective in (22) strictly improves by $\Delta z = \rho\,\mu$. Finally, $\mathbf{x}^0$ is updated to $\mathbf{x}^1$ as follows:

$$
\begin{array}{l}
\mathbf{x}_F^1 := \mathbf{x}_F^0 - \rho\,\bar{\mathbf{a}}_{P\omega} \\
\mathbf{x}_N^1 := \rho\,\mathbf{y}_N^0.
\end{array}
\tag{27}
$$

Table 1 The simplex tableau at $\mathbf{x}^0$

	x_1	x_2	x_3	λ_4	λ_5	λ_6	x_4	x_5	x_6	x_7	x_8			θ_ω
$\mathbf{c}$	2	3	1				10	17	-20	14	-4			-5
	1						2	2	1	-5	7	=	30	1
		1					4	3	-5	10	-10	=	25	-2
			1				-3	1	2	3	11	=	50	5
				1			0	0	6	5	-13	=	0	0
					1		0	0	3	4	-8	=	0	0
						1	0	0	3	-4	0	=	0	0
$\mathbf{x}^0$	30	25	50									$z^0 =$	185	
$\tilde{\mathbf{c}}$							-3	3	-9	-9	1			-6

2.4 Numerical example

Table 1 depicts a linear program in standard form comprising eight **x**-variables and six constraints. The degenerate solution $\mathbf{x}^0$ is already presented in the simplex tableau format: $(x_1^0, x_2^0, x_3^0) = (30, 25, 50)$ are the positive (or free) basic variables and the basis has been completed with artificial $\boldsymbol{\lambda}$-variables in rows 4, 5 and 6. The cost of this solution is $z^0 = 185$.

The vector $\mathbf{c}_F^\intercal = [2, 3, 1]$, equal to the dual vector for the top rows, is used for computing partial reduced cost vector $\tilde{\mathbf{c}}_N^\intercal = [-3, 3, -9, -9, 1]$. By inspection, we see that x_4 and x_5 are compatible with the row partition derived from the right-hand side values. One can observe that the associated columns are (trivial) combinations of the (unit) vectors of the free variables x_1, x_2 and x_3.

Both compatible variables would provide a nondegenerate pivot if chosen as entering variables but only x_4 has a negative partial reduced cost value $\tilde{c}_4 = -3$ (which is also equal to its reduced cost $\bar{c}_4$). The incompatible variables x_6 and x_7 possess a negative partial reduced cost value of –9 whereas $\tilde{c}_8 = 1$. The selection of incompatible variable x_6 or x_7 would result in a degenerate pivot while that of x_8 would increase the objective function by $1 \times (\frac{30}{7})$.

However, solving the pricing problem (25) over the last three rows results in a combination of the incompatible vectors with weights: $(y_6^0, y_7^0, y_8^0) = (0.4, 0.3, 0.3)$. This provides the compatible vector $\bar{\mathbf{a}}_\omega^\intercal = [1\ -2\ 5\ 0\ 0\ 0]$ for the variable θ_ω of reduced cost $\mu = -9(0.4) + -9(0.3) + 1(0.3) = -6$ and cost –5. The ratio test on the top three rows results in $\rho = \min\{\frac{30}{1}, -, \frac{50}{5}\} = 10$. The entering variable θ_ω takes value 10 and provides a strict improvement of the objective function of $-6 \times 10 = -60$. As a result, x_3 goes out of the basis, and other free variables x_1 and x_2 are respectively updated to 20 and 45. Alternatively, the variables x_6, x_7 and x_8 can be entered one by one in the basis, in any order, and this produces the same result. In the new solution of cost 125, the positive variables are $(x_1, x_2, \theta_\omega) = (20, 45, 10)$ or equivalently for $\mathbf{x}^1$ in terms of the original variables, $(x_1, x_2, x_6, x_7, x_8) = (20, 45, 4, 3, 3)$ while x_3, x_4 and x_5 are null variables.

From the five columns corresponding to the positive variables, the first five rows are independent and the artificial variable λ_6 is basic with a zero value in the last

Table 2 The basis **B** and its inverse at $\mathbf{x}^1$

$$\mathbf{B} = \left[\begin{array}{ccccc|c} 1 & & 1 & -5 & 7 & \\ & 1 & -5 & 10 & -10 & \\ & & 2 & 3 & 11 & \\ & & 6 & 5 & -13 & \\ & & 3 & 4 & -8 & \\ \hline & & 3 & -4 & 0 & 1 \end{array}\right] \quad \mathbf{B}^{-1} = \left[\begin{array}{ccccc|c} 1 & & -0.20 & -2.133 & 4.067 & \\ & 1 & 0.40 & 5.600 & -9.800 & \\ & & 0.08 & 0.453 & -0.627 & \\ & & 0.06 & -0.327 & 0.613 & \\ & & 0.06 & 0.007 & -0.053 & \\ \hline 0 & 0 & 0 & -2.667 & 4.333 & 1 \end{array}\right]$$

row. The inverse basis $\mathbf{B}^{-1}$ at $\mathbf{x}^1$ appears in Table 2 and is used to construct the next degenerate simplex tableau in Table 3.

$\mathbf{A}_{PF}^{-1}$, the inverse of the working basis within $\mathbf{B}^{-1}$, is used to compute the dual vector $\mathbf{c}_F^{\mathrm{T}}\mathbf{A}_{PF}^{-1} = [2, 3, -0.2, -1.133, 0.067]$ of the row set P and partial reduced costs $(\tilde{c}_3, \tilde{c}_4, \tilde{c}_5) = (1.2, -6.6, 4.2)$. Moreover,

$$\bar{\mathbf{a}}_{Zj} = -\mathbf{A}_{ZF}\mathbf{A}_{PF}^{-1}\mathbf{a}_{Pj} + \mathbf{a}_{Zj} = \mathbf{0}, \quad j \in \{3, 4, 5\},$$

characterizes column compatibility by computing

$$[\,\bar{\mathbf{a}}_{Z3}\,\bar{\mathbf{a}}_{Z4}\,\bar{\mathbf{a}}_{Z5}\,] = (0, 0, 0, -2.667, 4.333)[\,\mathbf{a}_{P3}\,\mathbf{a}_{P4}\,\mathbf{a}_{P5}\,] + [\,0\;0\;0\,] = [\,0\;0\;0\,].$$

The null variables x_3, x_4 and x_5 are compatible with the current row partition, and the optimal solution to the pricing problem at iteration 1 is $y_4^1 = 1$: x_4 enters the basis, being the only one with a negative reduced cost of –6.6. The ratio test on the top five rows results in $\rho = \min\{\frac{20}{2.6}, \frac{45}{2.8}, -, -, -\} = 7.692$ and the entering variable x_4 provides an objective function improvement of $-6.6 \times 7.692 = -50.769$. The variable x_1 goes out of the basis, and updated free variables x_2, x_6, x_7 and x_8 appear in Table 4, here presented in terms of the simplex tableau at $\mathbf{x}^2$ before being updated. Observe that the actual combination of variables x_6, x_7 and x_8 satisfies the last three rows at zero right-hand side. The cost of this solution is $z^2 = 74.231$.

$\mathbf{B}^{-1}$ for $\mathbf{x}^2$ appears in Table 5 from which $\mathbf{c}_F^{\mathrm{T}}\mathbf{A}_{PF}^{-1} = [-0.538, 3, 0.308, 4.282, -10.256]$ is computed and the partial reduced costs $(\tilde{c}_3, \tilde{c}_1, \tilde{c}_5) = (0.692, 2.538, 8.769)$. Since these are positive, $\mathbf{x}^2$ is optimal.

3 Linear algebra framework

To appreciate the generality of IPS, the reader is invited to consider its presentation only borrows from the algebraic manipulations of PS. The linear algebra framework

Table 3 The simplex tableau at $\mathbf{x}^1$

	x_1	x_2	x_6	x_7	x_8	λ_6	x_3	x_4	x_5		
$\mathbf{c}$	2	3	-20	14	-4		1	10	17		
	1						-0.20	2.60	1.80	=	20
		1					0.40	2.80	3.40	=	45
			1				0.08	-0.24	0.08	=	4
				1			0.06	-0.18	0.06	=	3
					1		0.06	-0.18	0.06	=	3
						1	0	0	0	=	0
$\mathbf{x}^1$	20	45	4	3	3					z^1 =	125
$\tilde{\mathbf{c}}$							1.2	-6.6	4.2		

Table 4 The simplex tableau at $\mathbf{x}^2$ before being updated

	x_4	x_2	x_6	x_7	x_8	λ_6	x_3	x_1	x_5		
c	10	3	-20	14	-4		1	2	17		
	2		1	-5	7			1	2	=	30
	4	1	-5	10	-10				3	=	25
	-3		2	3	11		1		1	=	50
			6	5	-13					=	0
			3	4	-8					=	0
			3	-4	0	1				=	0
$\mathbf{x}^2$	7.692	23.462	5.846	4.385	4.385					$z^2 =$	74.231
$\tilde{\mathbf{c}}$							0.692	2.538	8.769		

Table 5 The inverse basis $\mathbf{B}^{-1}$ at $\mathbf{x}^2$

$$\mathbf{B}^{-1} = \left[\begin{array}{ccccc|c} 0.385 & & -0.077 & -0.821 & 1.564 & \\ -1.077 & 1 & 0.615 & 7.897 & -14.179 & \\ 0.092 & & 0.062 & 0.256 & -0.251 & \\ 0.069 & & 0.046 & -0.474 & 0.895 & \\ 0.069 & & 0.046 & -0.141 & 0.228 & \\ \hline 0 & 0 & 0 & -2.667 & 4.333 & 1 \end{array}\right]$$

is put forth to derive another way to look at the row/column partition. Section 3.1 introduces the vector subspace $\mathbf{V}(\mathbf{A}_F)$ spanned by the column vectors of $\mathbf{A}_F$. This is followed in Sect. 3.2 by the practical use of an equivalent subspace basis $\mathbf{\Lambda}_f$. In Sect. 3.3, we examine a different subspace basis, $\mathbf{\Lambda}_r$, of possibly larger dimension $r \geq f$ that is sufficient to span $\mathbf{A}_F$. Section 3.4 discusses the pitfalls of this more general subspace basis and a modified algorithm is given in Sect. 3.5. For the record, the vector subspace notion is first mentioned in Benchimol et al. (2012) for the implementation of a stabilized DCA algorithm for the set partitioning problem.

3.1 Vector subspace $\mathbf{V}(\mathbf{A}_F)$

The concept of compatibility is contextual by nature since it assumes a row partition $\{P, \bar{P}\}$ of $\mathbf{A}_F$, where $\bar{P} = Z$. The reader might have observed that we have taken the liberty to omit this precision outside the definition of compatibility. It turns out that this omission works well in our favor. The following result holds the explanation Desrosiers et al. (2014) whereas Proposition 3 presents an alternative definition of compatibility which is impervious to the partition.

Proposition 2 *Let* $\mathbf{A}_{PF}$ *and* $\mathbf{A}_{QF}$ *be two working bases identifying different row partitions of* $\mathbf{A}_F$. *If vector* $\mathbf{a}$ *is compatible with partition* $\{P, \bar{P}\}$ *then it is also compatible with* $\{Q, \bar{Q}\}$. *Hence, we say* $\mathbf{a}$ *is compatible with* $\mathbf{A}_F$.

Proof Assume the vector $\mathbf{a}$ is compatible with the partition $\{P, \bar{P}\}$ and consider the following relation on set Q:

$$\mathbf{A}_{PF}^{-1}\mathbf{a}_P = \mathbf{A}_{QF}^{-1}\mathbf{a}_Q \quad \Longleftrightarrow \quad \mathbf{a}_Q - \mathbf{A}_{QF}\mathbf{A}_{PF}^{-1}\mathbf{a}_P = \mathbf{0}. \tag{28}$$

The right part is verified for every component $i \in Q$: true for $i \in Q \cap \bar{P}$ since the vector $\mathbf{a}$ is compatible whereas for $i \in Q \cap P$, $a_i - \mathbf{A}_{iF}\mathbf{A}_{PF}^{-1}\mathbf{a}_P = a_i - a_i = 0$. Hence,

$$\mathbf{a}_{\bar{Q}} - \mathbf{A}_{\bar{Q}F}\mathbf{A}_{QF}^{-1}\mathbf{a}_Q = \begin{cases} a_i - \mathbf{A}_{iF}\mathbf{A}_{PF}^{-1}\mathbf{a}_P & = a_i - a_i = 0 \quad \forall i \in \bar{Q} \cap P \\ a_i - \mathbf{A}_{iF}\mathbf{A}_{PF}^{-1}\mathbf{a}_P & = 0 \quad \forall i \in \bar{Q} \cap \bar{P}, \end{cases} \tag{29}$$

the last equality being true since the vector $\mathbf{a}$ is compatible with the partition $\{P, \bar{P}\}$. □

Proposition 3 A vector $\mathbf{a} \in \mathbb{R}^m$ (and the associated variable, if any) is *compatible with* $\mathbf{A}_F$ if and only if it belongs to $\mathbf{V}(\mathbf{A}_F)$.

Proof We first show that if Definition 1 is satisfied for some partition $\{P, Z\}$ then the statement rings true. We then show that the converse is also true. Assume that the vector $\mathbf{a}$ is compatible such that $\bar{\mathbf{a}}_Z = \mathbf{a}_Z - \mathbf{A}_{ZF}\mathbf{A}_{PF}^{-1}\mathbf{a}_P = \mathbf{0}$. Let $\boldsymbol{\alpha} := \mathbf{A}_{PF}^{-1}\mathbf{a}_P$. Then, $\begin{bmatrix} \mathbf{a}_P \\ \mathbf{a}_Z \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{PF}\boldsymbol{\alpha} \\ \mathbf{A}_{ZF}\boldsymbol{\alpha} \end{bmatrix}$ meaning that the vector $\mathbf{a}$ indeed belongs to $\mathbf{V}(\mathbf{a}_F)$. Let us now assume that there exists some $\boldsymbol{\alpha} \in \mathbb{R}^f$ such that the vector $\mathbf{a}$ is a linear combination of the column vectors of $\mathbf{A}_F$. Since $\mathbf{A}_F$ is a subspace basis, there exists some row set P such that $\mathbf{A}_{PF}$ is invertible. Then, $\boldsymbol{\alpha} = \mathbf{A}_{PF}^{-1}\mathbf{a}_P$ and compatibility of the vector $\mathbf{a}$ follows. □

A consequence of Proposition 3 is that every subset $\boldsymbol{\Lambda}_f$ of f independent vectors of $\mathbf{V}(\mathbf{A}_F)$ can be used as a subspace basis for $\mathbf{V}(\mathbf{A}_F)$. Let us explicitly recall the definition of a vector basis as a linearly independent spanning set. A simple but important observation is the following: The set of f independent vectors of $\mathbf{A}_F$ identified in IPS is therefore a *minimal* spanning set capable of representing the current solution, $\mathbf{A}_F\mathbf{x}_F^0 = \mathbf{b}^0$. Indeed, the very construction of the working basis in $\mathbf{B}$ implies that $\mathbf{A}_F$ spans $\mathbf{b}^0$, that is, $\mathbf{x}_F^0 = \mathbf{A}_{PF}^{-1}\mathbf{b}_P^0$, see the system of linear equations in (6) or (8).

3.2 Subspace basis $\boldsymbol{\Lambda}_f$

The identification of the working basis is one of the bottleneck operations of IPS. Furthermore, as the reader can observe from formulation (8), it is useless to multiply by $\mathbf{A}_{PF}^{-1}$ the rows in set P to identify the improving variable $\theta_\omega, \omega \in \Omega$, if any. Indeed, only $\bar{\mathbf{a}}_{P\omega}$ needs to be computed to perform the ratio test (16). An

alternative set to $\mathbf{A}_F$ of f independent vectors that spans $\mathbf{V}(\mathbf{A}_F)$ is $\mathbf{\Lambda}_f = \begin{bmatrix} \mathbf{I}_f \\ \mathbf{M} \end{bmatrix}$, where $\mathbf{M} = \mathbf{A}_{ZF}\mathbf{A}_{PF}^{-1}$. Together with $\mathbf{\Lambda}_f^{\perp} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_{m-f} \end{bmatrix}$, it provides basis $\mathbf{T} := [\mathbf{\Lambda}_f, \mathbf{\Lambda}_f^{\perp}]$ of $\mathbb{R}^m$ and its inverse:

$$\mathbf{T} = \begin{bmatrix} \mathbf{I}_f & \mathbf{0} \\ \mathbf{M} & \mathbf{I}_{m-f} \end{bmatrix} \quad \text{and} \quad \mathbf{T}^{-1} = \begin{bmatrix} \mathbf{I}_f & \mathbf{0} \\ -\mathbf{M} & \mathbf{I}_{m-f} \end{bmatrix}. \tag{30}$$

The LP formulation obtained after the change of variables and the transformation by the more practical $\mathbf{T}^{-1}$ results in an equivalent system for which only the rows in set Z are transformed:

$$\begin{array}{lllllll} z^{\star} = & \mathbf{c}_L^{\intercal}\mathbf{x}_L^0 + \mathbf{c}_U^{\intercal}\mathbf{x}_U^0 + & \min & \mathbf{c}_F^{\intercal}\mathbf{x}_F & + & \mathbf{d}_N^{\intercal}\mathbf{y}_N & \\ & & \text{s.t.} & \mathbf{A}_{PF}\mathbf{x}_F & + & \mathbf{A}_{PN}^0\mathbf{y}_N & = \mathbf{b}_P^0, \quad [\boldsymbol{\psi}_P] \\ & & & & & \bar{\mathbf{A}}_{ZN}^0\mathbf{y}_N & = \mathbf{0}, \quad [\boldsymbol{\psi}_Z] \\ & & & \mathbf{l}_F \le \mathbf{x}_F \le \mathbf{u}_F, & & \mathbf{0} \le \mathbf{y}_N \le \mathbf{r}_N, & \end{array} \tag{31}$$

where $\bar{\mathbf{A}}_{ZN}^0 = \mathbf{A}_{ZN}^0 - \mathbf{M}\mathbf{A}_{RN}^0$. Similarly to (9) and (10), $\boldsymbol{\pi}$ can be retrieved from the dual vector $\boldsymbol{\psi}$ in (31) using the expression $\boldsymbol{\pi}^{\intercal} = \boldsymbol{\psi}^{\intercal}\mathbf{T}^{-1}$:

$$[\boldsymbol{\pi}_P^{\intercal}, \boldsymbol{\pi}_Z^{\intercal}] = \left[\boldsymbol{\psi}_P^{\intercal} - \boldsymbol{\psi}_Z^{\intercal}\mathbf{M},\ \boldsymbol{\psi}_Z^{\intercal}\right]. \tag{32}$$

When all is said and done, using vector subspace properties enables one to derive a working basis using any and all efficient methods to extract an equivalent subspace basis. Furthermore, depending on the application, the inverse is implicitly obtained in $\mathbf{M}$ as a by-product of the decomposition. Of course, having access to the LP solver's own LU-decomposition would be quite practical. Note that although $\mathbf{B}$ constructed in (5) is implicitly considered as a simplex basis in IPS, $\mathbf{T}$ is more generally defined as a basis in $\mathbb{R}^m$, that is, an invertible linear transformation in $\mathbb{R}^m$.

3.3 Subspace basis $\mathbf{\Lambda}_r, r \ge f$

Let us consider the general situation where r, the dimension of the subspace basis $\mathbf{\Lambda}_r$ spanning the columns of $\mathbf{A}_F$, is larger than or equal to f, the number of free variables. Assume $\mathbf{\Lambda}_r$ includes the f columns of $\mathbf{A}_F$ and $r - f \ge 0$ additional columns such that these r columns are linearly independent. Using a restricted phase I, one identifies r independent rows in subset $R \subseteq \{1, \ldots, m\}$ and the subspace basis can take the form $\mathbf{\Lambda}_r = \begin{bmatrix} \mathbf{I}_r \\ \mathbf{M} \end{bmatrix}$, where $\mathbf{M}$ is an $(m-r) \times r$ matrix, whereas $\mathbf{\Lambda}_r^{\perp} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_{m-r} \end{bmatrix}$. Let $\mathbf{V}(\mathbf{\Lambda}_r)$ be the vector subspace spanned by $\mathbf{\Lambda}_r$. At the end of the day, the definition of compatibility can be enlarged to the spanning set of the chosen subspace basis.

Definition 2 A vector $\mathbf{a} \in \mathbb{R}^m$ (and the associated variable, if any) is *compatible with* $\mathbf{\Lambda}_r$ if and only if it belongs to $\mathbf{V}(\mathbf{\Lambda}_r)$.

3.4 Words of caution about compatibility

Once the general form of the subspace basis $\mathbf{\Lambda}_r$ is retained, it is delicate to still claim this modified version as IPS. If the latter can be seen as a *poorer* vector subspace which obviously includes $\mathbf{A}_F$, the added granularity provided by the superfluous columns yields a denser compatible set.

The danger of over-spanning $\mathbf{A}_F$ is that a compatible surrogate variable $\theta_\omega, \omega \in \mathbf{\Omega}$, found by the pricing problem does not guarantee a strictly improving pivot. Indeed, any value $\bar{\mathbf{a}}^0_{i\omega} \neq 0, i \in R$, corresponding to $\bar{b}^0_i = 0, i \in R$, is potential cause for a zero step size, hence a degenerate pivot. Observe that the magnitude of the value $\bar{\mathbf{a}}^0_{i\omega}$ is irrelevant, what really matters is its sign. In a very superficial sense, the probability of making a degenerate pivot thus increases exponentially by one half for every extra row, i.e., $1 - (1/2)^{r-f}$. When $r > f$, the probability ranges from one half to almost surely very rapidly. For that matter, even an incompatible variable might induce a nondegenerate pivot with probability $(1/2)^{m-f}$. The reader is invited to take a look at the variable x_8 in the numerical example of Sect. 2.4 to be convinced of the nondegenerate potential. Of course, a more refined analysis of the probabilities would include the configuration of matrix $\mathbf{A}$ and at this point falls outside the purpose of this paper.

In most if not all literature surrounding IPS and its derivatives, the concept of compatibility is associated with a nondegenerate pivot. While it is true that in the purest form of IPS, a compatible variable necessarily induces a nondegenerate pivot, the implication of the previous paragraph denies this synonymy for the general form of the subspace basis, as it stands the implemented version. What does this all mean? The linear algebra framework that surrounds IPS provides a more robust definition of compatibility. As the latter gains in flexibility, it loses in certainty. Compatibility provides a way to categorize variables by their capacity to induce nondegenerate pivots with fair accuracy. We guess researchers have taken the liberty to address one for the other because of the intent behind the partition scheme. A leap of faith comes to mind.

To sum up, this larger subspace basis $\mathbf{\Lambda}_r$ breaks away from the strictly improving pivot construction of IPS. It is however a necessary evil that gives a lot of freedom in the implementation and, more importantly, closes the theoretical gap between IPS and DCA.

3.5 Modified IPS algorithm

Figure 2 contains the modifications necessary to include the linear algebra framework to the vanilla version of IPS. In Step 1, the construction of the working basis uses the representation $\mathbf{T}$. For Step 2, the compatible set is constructed with the alternative Definition 2. Steps 3 and 4 rely on the modified row partition $\{R, \bar{R}\}$,

but their essence remains otherwise untouched and therefore see no particular caveat. Neither does Step 5.

4 Aiming for efficiency

This section serves the practical side of an implementation of IPS. The fourth step of IPS, namely the exchange mechanism, brings the solution of the improved pricing step back to what can be called a *control system*. This is indeed where feasibility is maintained by using the flexibility of the free variables and the interval bounds otherwise omitted from the pricing step. With that being said, this process of information sharing between the pricing step and the control system is quite close to a master problem/subproblem paradigm. In fact, with a better understanding of the pricing step, we argue in Sect. 4.1 that IPS corresponds to dynamically applying a Dantzig–Wolfe reformulation Dantzig and Wolfe (1960) at every iteration, the row partition being done according to the current solution vector $\mathbf{x}^0$ given by $[\mathbf{x}^0_F; \mathbf{x}^0_L; \mathbf{x}^0_U]$.

This interpretation of IPS can result in very flexible resolution strategies. Among these is the usage of the convex combination $\omega \in \Omega$ and its surrogate variable θ_ω opposed to the column components of ω and their respective original $\mathbf{x}$-variables. We also know from column generation that generating several columns during one iteration of the pricing step is highly efficient. In line with this idea also comes that of using heuristics to solve the pricing problem during the early stage of the resolution process. The fourth and perhaps most important idea defers to the time consuming task of updating the row partition. Such is the content of the three subsequent subsections (Sects. 4.2, 4.3 and 4.4) which examine these various ways to accelerate IPS. Section 4.5 presents the dynamic Dantzig–Wolfe implementation of IPS while Section 4.6 shares computational results gathered from different papers.

4.1 Dynamic Dantzig–Wolfe decomposition

We now present an interpretation of IPS in terms of a decomposition scheme proposed by Metrane et al. (2010) for standard linear programs. Here is an adaptation for the bounded case.

Consider a Dantzig–Wolfe decomposition of the previous so-called compact formulation (31) which has a block angular structure. The equality constraints in

1 Generic basis $\mathbf{T}$, transformation $\mathbf{T}^{-1}$, row partition $\{R, \bar{R}\}$ of $\boldsymbol{\Lambda}_r$;
2 Compatibility with the row partition $\{R, \bar{R}\}$ of $\boldsymbol{\Lambda}_r$ <optional>;
3 Improved pricing step: optimize the minimum reduced cost μ;
4 Exchange mechanism from $\mathbf{x}^0$ to $\mathbf{x}^1$;
5 Update the column partition $\{F, L, U\}$ and goto Step 1;

Fig. 2 Modified IPS algorithmic steps

set P together with the interval constraints $\mathbf{l}_F \leq \mathbf{x}_F \leq \mathbf{u}_F$ and upper bounds $\mathbf{y}_N \leq \mathbf{r}_N$ stay in the master problem structure. The equality constraints in the row set Z and nonnegativity constraints $\mathbf{y}_N \geq \mathbf{0}$ form the subproblem domain:

$$\mathcal{SP} := \{\mathbf{y}_N \geq \mathbf{0} \mid \bar{\mathbf{A}}^0_{ZN}\mathbf{y}_N = \mathbf{0}\}. \tag{33}$$

The Dantzig–Wolfe decomposition builds on the representation theorems by Minkowski and Weyl (see Schrijver 1986; Desrosiers and Lübbecke 2011) that any vector $\mathbf{y}_N \in \mathcal{SP}$ can be reformulated as a convex combination of extreme points plus a nonnegative combination of extreme rays of $\mathcal{SP}$. Moreover, $\mathcal{SP}$ is a cone for which the only extreme point is the null vector $\mathbf{y}_N = \mathbf{0}$ at zero cost. Since this extreme point does not contribute to the master problem constraints, it can as such be discarded from the reformulation. Vector $\mathbf{y}_N$ can thus be expressed as a nonnegative combination of the extreme rays $\{\mathbf{y}^\omega_N\}_{\omega\in\Omega}$:

$$\mathbf{y}_N = \sum_{\omega\in\Omega} \mathbf{y}^\omega_N \theta_\omega, \ \theta_\omega \geq 0, \quad \forall \omega \in \Omega.$$

Substituting in the master problem structure, LP becomes:

$$\begin{array}{lllll} z^\star = \mathbf{c}_L^\intercal \mathbf{x}^0_L + \mathbf{c}_U^\intercal \mathbf{x}^0_U + & \min & \mathbf{c}_F^\intercal \mathbf{x}_F & + & \sum_{\omega\in\Omega}[\mathbf{d}_N^\intercal \mathbf{y}^\omega_N]\theta_\omega \\ & \text{s.t.} & \mathbf{A}_{PF}\mathbf{x}_F & + & \sum_{\omega\in\Omega}[\mathbf{A}^0_{PN}\mathbf{y}^\omega_N]\theta_\omega \quad = \mathbf{b}^0_P, \quad [\boldsymbol{\psi}_P] \\ & & & & \sum_{\omega\in\Omega}[\mathbf{y}^\omega_N]\theta_\omega \quad \leq \mathbf{r}_N, \\ & & \mathbf{l}_F \leq \mathbf{x}_F \leq \mathbf{u}_F, & & \theta_\omega \geq 0, \forall \omega \in \Omega. \end{array} \tag{34}$$

At any iteration of IPS, none of the θ-variables are yet generated and the inequality constraints in (34) are not binding. Therefore, the dual vector for these constraints is null and the reduced cost of variable $\theta_\omega, \omega \in \Omega$, is given by:

$$[\mathbf{d}_N^\intercal \mathbf{y}^\omega_N] - \boldsymbol{\psi}_P^\intercal [\mathbf{A}^0_{PN}\mathbf{y}^\omega_N] = (\mathbf{d}_N^\intercal - \boldsymbol{\psi}_P^\intercal \mathbf{A}^0_{PN})\mathbf{y}^\omega_N = \tilde{\mathbf{d}}_N^\intercal \mathbf{y}^\omega_N,$$

where $\tilde{\mathbf{d}}_N$ is the partial reduced cost vector already used in IPS, see formulation (12). Now, any negative reduced cost ray in $\mathcal{SP}$ results in the same subproblem minimum objective value, that is, $-\infty$. However, observe that for any nonzero solution in the cone defined by $\mathcal{SP}$ in (33), there exists a scaled one such that $\mathbf{1}^\intercal \mathbf{y}_N = 1$. Therefore, without loss of generality, the domain of the subproblem can be rewritten as

$$\mathcal{SP}_N := \{\mathbf{y}_N \geq \mathbf{0} \mid \bar{\mathbf{A}}^0_{ZN}\mathbf{y}_N = \mathbf{0}, \ \mathbf{1}^\intercal \mathbf{y}_N = 1\}. \tag{35}$$

Hence, an equivalent subproblem in this Dantzig–Wolfe decomposition, searching for a negative reduced cost column until optimality is reached, is exactly the one defined by the primal pricing problem (13) in IPS:

$$\min \ \tilde{\mathbf{d}}_N^\intercal \mathbf{y}_N \quad \text{s.t.} \quad \mathbf{y}_N \in \mathcal{SP}_N. \tag{36}$$

The bottleneck of this algorithm is the improved pricing step. Recall that the content of the latter is a ripple effect of the decomposition choice. These ideas can therefore be separated in two categories: the first supports the idea that the master problem

and the pricing step are communicating vessels, the second is solely aimed at the pricing step in an effort to find solutions, not necessarily optimal, faster. Before moving on to the three subsections which examine the aforementioned various ways to accelerate IPS, let us recall how the master problem may be fed with surrogate variables or their original column vector content.

4.1.1 Convex combination vs. column components

The weights $\mathbf{y}_N^0$ dictate the content of convex combination ω and ascertain the compatibility requirement of the entering variable θ_ω. By neglecting these weights, the original column components of the convex combination can be fed directly to the compact formulation (31) along with their associated original $\mathbf{x}$-variables. While this certainly seems counterproductive for the one direction, let us go through the mechanics for the sake of argument. Discarding the weights also implies that the compatible faith of this group of columns is lost. The active constraints in the pricing problem must therefore also be passed to the compact formulation. The latter can then obviously be solved to $\mathbf{x}_N = \mathbf{y}_N^0 \theta_\omega$, a process that leads to the same objective value as would pivoting θ_ω. The column components mechanics has the potential to shine when one thinks of a column generation framework where multiple columns are brought back to the restricted master problem. In this perspective, the original column components fed to the compact formulation could be arranged with their siblings from other directions at different levels thus granting more freedom than the surrogate variables provide. Similar techniques to solve large-scale linear multi-commodity flow problems were previously used by Löbel (1998) and Mamer and McBride (2000), whereas Valério de Carvalho (1999, 2002) propose a network-based compact formulation of the cutting stock problem in which the classical knapsack subproblem is solved as a shortest path problem. In all these applications, the compact formulation is written in terms of arc flow variables. When a subproblem generates a *path* with a negative reduced cost, the *arcs* of this path are iteratively added to the compact formulation. This process allows the implicit combination of arcs into paths without having to generate these. Sadykov and Vanderbeck (2013) describe this in generality.

4.2 Subspace basis update

Postponing the subspace basis update can be taken advantage of on two fronts: before and after updating to the new solution $\mathbf{x}^1$. On the first front, it is indeed a basic idea to harvest more information from the pricing problem than the one iteration. Let this agenda be known as *multiple improving directions*. We present two specific scenarios before the general one. The first scenario is the particular case of *independent improving directions* while the second is the *compatible restricted master problem*. On the second front, from the Dantzig–Wolfe mindset, it becomes clear that entering an improving variable θ_w in the master problem (34) does not necessarily warrant an update of the subspace basis. In either case, it is in effect a

matter of manipulating the dual variables. The price to pay is the possibility of making degenerate pivots on some of these directions.

4.2.1 Independent improving directions

IPS relies on the strictly improving property of the algorithm to guarantee that the exchange mechanism goes through the components of θ_ω with a strictly positive step size, see (16). If two variables θ_{ω_1} and θ_{ω_2} can be identified from the pricing problem such that compatibility is obtained from orthogonal vectors of $\mathbf{V}(\mathbf{A}_F)$, then $\bar{\mathbf{a}}^0_{\omega_1}$ and $\bar{\mathbf{a}}^0_{\omega_2}$ are independent from each other and can be added to the current solution in any order both yielding a predictable improvement. Independence constraints need not be added to the pricing problem in order to carry out this strategy, it suffices to remove variables that already *contribute* in the first direction of the variable θ_{ω_1}. Indeed, the selection of columns the latter contains should of course be removed from the pricing problem. Among themselves and variables of $\mathbf{x}_F$ used to complete the direction, these columns have nonzero elements on several rows, i.e., they contribute on each of these rows. Any variable that sports a nonzero value on any of these same rows shares a contribution and can therefore be removed from the pricing problem. In other words, removing every variable that contributes to the aforementioned rows amounts, for all intents and purposes, to discarding these constraints as well.

4.2.2 Compatible restricted master problem (RMP_{FC})

Consider the row partition $\{R, Z\}$ of $\mathbf{\Lambda}_r$, where $Z = \bar{R}$. By Definition 2, the columns of $\mathbf{A}_F$ are compatible with $\mathbf{\Lambda}_r$. Denote by $\mathbf{A}^0_C$, $C \subseteq N$, the columns of $\mathbf{A}^0_N$ compatible with $\mathbf{\Lambda}_r$. Any of these can easily be identified in $O(m)$ time using PE, see Sect. 7. Let $\mathbf{A}_I$ be the incompatible columns, $I := N \setminus C$. Using $\mathbf{T} = [\mathbf{\Lambda}_r, \mathbf{\Lambda}_r^\perp]$ as a basis of $\mathbb{R}^m$ and applying the transformation $\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{I}_r & \mathbf{0} \\ -\mathbf{M} & \mathbf{I}_{m-r} \end{bmatrix}$ on the formulation (3), we have $\bar{\mathbf{a}}^0_{Zj} = \bar{\mathbf{b}}^0_Z = \mathbf{0},\ \forall j \in F \cup C$. Let $\bar{\mathbf{A}}^0_{ZI} := \mathbf{A}^0_{ZI} - \mathbf{M}\mathbf{A}^0_{RI}$. LP becomes

$$
\begin{array}{lrcccll}
z^\star = \mathbf{c}_L^\intercal \mathbf{x}_L^0 + \mathbf{c}_U^\intercal \mathbf{x}_U^0 + \min & \mathbf{c}_F^\intercal \mathbf{x}_F & + & \mathbf{d}_C^\intercal \mathbf{y}_C & + & \mathbf{d}_I^\intercal \mathbf{y}_I & \\
\text{s.t.} & \mathbf{A}_{RF}\mathbf{x}_F & + & \mathbf{A}^0_{RC}\mathbf{y}_C & + & \mathbf{A}^0_{RI}\mathbf{y}_I & = \mathbf{b}^0_R, \quad [\boldsymbol{\psi}_R] \\
 & & & & & \bar{\mathbf{A}}^0_{ZI}\mathbf{y}_I & = \mathbf{0}, \quad [\boldsymbol{\psi}_Z] \\
 & \mathbf{l}_F \le \mathbf{x}_F \le \mathbf{u}_F, & & \mathbf{0} \le \mathbf{y}_C \le \mathbf{r}_C, & & \mathbf{0} \le \mathbf{y}_I \le \mathbf{r}_I. &
\end{array}
\tag{37}
$$

Restricting the formulation (37) to columns in the set $F \cup C$ (the compatible variables) yields the compatible restricted master problem RMP_{FC} defined on the row set R, much easier to solve than (1) as it involves fewer variables and, more importantly, fewer constraints. As such, it is less subject to degeneracy.

Of course, its optimal solution does not necessarily solve LP. It is equivalent to having exhausted the pricing step of all improving compatible variables one iteration after another without having updated the subspace basis. Whether one

should update the latter now or retrieve more directions from the pricing step is arguably the exact same question as one would face after the *first* direction is retrieved from the complete pricing step.

4.2.3 Multiple improving directions

To have access to valid combinations of incompatible columns in $\mathbf{A}_I^0$, the Dantzig–Wolfe decomposition obtained from $\mathbf{x}^0$ is maintained. Keeping in the subproblem the equalities from the row set Z, the nonnegativity requirements $\mathbf{y}_I \geq \mathbf{0}$, and a scaling constraint on $\mathbf{y}_I$ leads to the following formulation.

$$\min\ \tilde{\mathbf{d}}_I^{\intercal}\mathbf{y}_I \quad \text{s.t.} \quad \mathbf{y}_I \in \mathcal{SP}_I := \left\{\mathbf{y}_I \geq \mathbf{0} \mid \bar{\mathbf{A}}_{ZI}^0\mathbf{y}_I = \mathbf{0},\ \mathbf{1}^{\intercal}\mathbf{y}_I = 1\right\}. \tag{38}$$

As previously derived in Sect. 4.1, the substitution of the extreme rays generated from $\mathcal{SP}_I$, $\mathbf{y}_I^{\omega}, \omega \in \Omega$, into the master problem gives

$$\begin{array}{rlllll} z^{\star} = \mathbf{c}_L^{\intercal}\mathbf{x}_L^0 + \mathbf{c}_U^{\intercal}\mathbf{x}_U^0 + \min & \mathbf{c}_F^{\intercal}\mathbf{x}_F & + & \mathbf{d}_C^{\intercal}\mathbf{y}_C & + & \sum_{\omega\in\Omega}[\mathbf{d}_I^{\intercal}\mathbf{y}_I^{\omega}]\theta_{\omega} \\ \text{s.t.} & \mathbf{A}_{RF}\mathbf{x}_F & + & \mathbf{A}_{RC}^0\mathbf{y}_C & + & \sum_{\omega\in\Omega}[\mathbf{A}_{RI}^0\mathbf{y}_I^{\omega}]\theta_{\omega} \quad = \mathbf{b}_R^0, \quad [\boldsymbol{\psi}_R] \\ & & & & & \sum_{\omega\in\Omega}[\mathbf{y}_I^{\omega}]\theta_{\omega} \quad \leq \mathbf{r}_N, \\ & \mathbf{l}_F \leq \mathbf{x}_F \leq \mathbf{u}_F, & & \mathbf{0} \leq \mathbf{y}_C \leq \mathbf{r}_C, & & \theta_{\omega} \geq 0, \forall\omega \in \Omega, \end{array} \tag{39}$$

and similarly to (9)–(10) or (32), the dual vector $\boldsymbol{\pi}$ can be retrieved using the expression $\boldsymbol{\pi}^{\intercal} = \boldsymbol{\psi}^{\intercal}\mathbf{T}^{-1}$:

$$[\boldsymbol{\pi}_R^{\intercal}, \boldsymbol{\pi}_Z^{\intercal}] = \left[\boldsymbol{\psi}_R^{\intercal} - \boldsymbol{\psi}_Z^{\intercal}\mathbf{M},\ \boldsymbol{\psi}_Z^{\intercal}\right]. \tag{40}$$

Ranging from heuristically modifying the dual variables to discarding certain $\mathbf{y}$-variables or type of solutions, extracting many interesting directions from the pricing problem is then a matter of creativity. Notice that using all the compatible variables in RMP_{FC} is one such heuristic.

4.2.4 Postponing subspace basis update past $\mathbf{x}^1$

Once again, the row partition is only the fruit of a linear transformation $\mathbf{T}^{-1}$ at a given iteration. We argue that using surrogate variables allows to reuse the previous subspace basis because it is simply the result of the particular Dantzig–Wolfe decomposition partitioning rows into $\{R, \bar{R}\}$. Unfortunately, when more than one former free variable becomes degenerate, the old subspace basis now spans degenerate basic variables. It is therefore possible to maintain the old subspace basis but it implies the use of the more general form $\boldsymbol{\Lambda}_r$. In accordance with Sect. 3.3, we state that an update is in order when the actual number of free variables $|F|$ is relatively different from $|R|$, the row size of the master problem.

4.3 Vector subspace flexibility

Given that the vector subspace is defined with respect to the matrix of free variables $\mathbf{A}_F$, this section shows that it is even possible to play with the set of free

variables as we see fit. The first trick cheats the free status with algebraic manipulations while the other considers a particular type of upper bounds.

4.3.1 Coerced degeneracy

Another highly important concept is that of *coerced* degeneracy. This is used in the capacitated minimum cost network flow problem which can artificially render any current free variable into two degenerate ones on the residual network, see Ahuja et al. (1993). Indeed, an arc variable x_{ij} taking a value $\ell_{ij} < x^0_{ij} < u_{ij}$ on the original network formulation can be replaced by two variables representing upwards $(0 \leq y_{ij} \leq u_{ij} - x^0_{ij})$ and downwards $(0 \leq y_{ji} \leq x^0_{ij} - \ell_{ij})$ possible flow variations. The current values of these **y**-variables is null and again this can modify the relative row sizes of the master and the pricing problems. On either count, the choice of the vector subspace results in a degenerate free pricing step.

4.3.2 Implicit upper bounds

Some applications have a structure that *implicitly* bounds some variables by the sheer force of the technological constraints. For instance, the assignment and the set partitioning models have such a feature. As a matter of fact, all variables in both of these problems are bounded above by 1, yet the explicit bound needs not be added to the formulation. That is to say that a variable x_j features an *implicit* upper bound u_j if $x_j > u_j$ is infeasible regardless of the values of the remaining variables.

Taking upper bounds into account is an obligatory path to guarantee strictly improving directions. We argue that, in presence of implicit upper bounds, IPS can be applied in two different manners with respect to the way these upper bounds are taken into account. In the first case, upper bounds are stated in the formulation whereas the second case omits them altogether. Assume the variable $x_j = u_j$ has reached its implicit upper bound. In the explicit formulation, when an upper bound is reached, it is taken into account thus sending x_j in the pricing problem. In the silenced formulation, the variable $x_j = u_j$ is *assumed* to be free.

Since the bound is implicit, it should be obvious that both pricing problems should yield only nondegenerate directions. It is trivial in the first case since the upper bound is explicitly taken into account. In the second case, it must be shown that the pricing problem cannot identify a direction that would increase the variable x_j. The fact that $x_j \leq u_j$ is implicit from the set of technological constraints translates into the coefficients of the θ_ω-variables in (14) as these can be derived from the Dantzig–Wolfe reformulation in (34), *equivalent to the original formulation*. Therefore, one finds the following equality constraint for x_j when it reaches its implicit upper bound:

$$x_j + [\bar{a}^0_{j\omega}]\theta_\omega = u_j,$$

where $\theta_\omega \geq 0,\ \forall \omega \in \Omega$. Since $x_j \leq u_j$, coefficients $[\bar{a}^0_{j\omega}] \geq 0,\ \forall \omega \in \Omega$, and the *assumed* free variable x_j can only decrease during the exchange mechanism.

The main difference between these two choices echoes the vector subspace $\mathbf{V}(\mathbf{A}_F)$ and thus the set of compatible variables, see Proposition 3. Consider vector subspaces spanned by $\mathbf{A}_F$ which contains or not implicit bounded variables. The distinction lies in the compatibility set and different $\mathbf{A}_F$ modify the relative row sizes of the master (f) and the subproblem ($m - f + 1$). The added granularity provided by the additional vectors in the first case creates a denser linear span and thus allows more variable compatibility.

Observe that variables in N are always treated correctly by the pricing problem since they are observably at one of their bounds. The extension of this result is that a variable in N could be *assumed* to be free, if it can be shown that its current value is an implicit upper bound. While some very preliminary results are available in Sect. 4.6 with respect to implicit bounds formulations, both the theoretical and practical implications have yet to be explored meaningfully. This concept even has an impact within the scope of PS; a variable at its implicit bound could be either basic or nonbasic in the former case whereas it would necessarily be basic in the second.

4.4 Partial pricing

In this subsection, we discuss possible partial pricing choices to accelerate the resolution process of the pricing step without compromising optimality. That is, as long as the last iteration uses the complete model, intermediate pricing steps can use heuristic notions. Partial pricing strategies become appealing in diversified aspects. For example, one can use various subsets of compatible and incompatible variables to reduce the density of the pricing problem. We present three such biases: partial cost, residual capacity, and rank-incompatibility. These ideas can of course be mixed as deemed worthy.

4.4.1 Partial cost bias

An incompatible variable $j \in N$ can be temporarily discarded if its partial reduced cost $\tilde{d}_j$ is greater than some threshold value, the most simple one being zero.

4.4.2 Residual capacity bias

The idea of this bias is to guarantee a minimum step size ρ. One look at the modified ratio test (16) suffices to see that the residual capacity bias also involves free variables. On the one hand, we want to keep the variable $j \in N$ if its residual capacity r_j is relatively large. On the other hand, the coerced degeneracy principle must be used on free variables to keep only those where both values $\bar{b}^0_i - l_i$ and $u_i - \bar{b}^0_i, i \in R$ are large. Since the ratio test also depends on $\bar{a}^0_{i\omega}$, it makes this

guarantee all the more difficult to appreciate on arbitrary matrices $\mathbf{A}$. Nevertheless, the idea works well when it is embedded in the minimum mean cycle-canceling algorithm, an extreme case of row partition where $|F| = 0$ since coerced degeneracy is applied on *all* free variables. Observe that once the coerced degeneracy is applied, it might be possible to keep one of the coerced free variables in the pricing problem.

4.4.3 Rank-incompatibility bias

Another possibility is to define the pricing step against *rank-incompatibility*. This means that the incompatible variables are attributed a rank according to the degree of incompatibility they display. The pricing problem sequentially uses lower rank incompatible variables. Intuitively, the point is not to stray too far from the current compatibility definition and thus limit the perturbation caused by modifying it. This concept is first seen under the name Multi-phase DCA (MPDCA) in the paper of Elhallaoui et al. (2010).

4.5 Dynamic Dantzig–Wolfe algorithm

Figure 3 presents the implemented version of IPS inspired by the dynamic Dantzig–Wolfe construction. The first two steps recuperate the linear algebra work. The biggest modification thus entails the work done in Step 3 which is now broken down into smaller components. The first utilizes RMP_{FC} by solving a row-reduced master problem with only compatible variables. The second calls the pricing problem where dual multipliers are updated and only incompatible variables remain. The latter can be solved several times to retrieve many possibly improving directions using whatever arsenal available to the user to accomplish said task. Finally, the exchange mechanism in Step 5 can be applied to every predetermined direction, yet it is simpler to let a PS code create a new working basis using all the gathered information simultaneously. The reason for this are threefold. First, it ensures that the solution $\mathbf{x}^1$ is basic. Second, it fetches updated dual multipliers for the row set R in case the generic basis is not updated. Third, it allows for a possibly better solution than the sequential work. With this new solution $\mathbf{x}^1$, the algorithm loops and the partition may (goto Step 1) or may not (goto Step 3b) (in the surrogate variable environment) be updated. The pricing step will be influenced by new dual variables either way.

1 Generic basis $\mathbf{T}$, transformation $\mathbf{T}^{-1}$, row partition $\{R, \bar{R}\}$ of $\boldsymbol{\Lambda}_r$;
2 Compatibility with the row partition $\{R, \bar{R}\}$ of $\boldsymbol{\Lambda}_r$ <optional>;
3a Restricted master problem: solve RMP_{FC} to optimality <optional goto Step 5>;
3b Improved pricing step: optimize the minimum reduced cost μ <optional repeat>;
4 Exchange mechanism from $\mathbf{x}^0$ to $\mathbf{x}^1$;
5 Update the column partition $\{F, L, U\}$ and goto Step 1 <optional goto Step 3b instead>;

Fig. 3 Dynamic Dantzig–Wolfe algorithmic steps

4.6 Computational results for IPS

As the reader might have guessed, these ideas must be meticulously handled in the practical implementation of IPS for it to be competitive. The computational results for the latter have therefore been deferred to this point. Linear programs in standard form have been used for the comparison between IPS and CPLEX's PS. The two main ideas used to obtain these results are the subspace basis update along with the compatible restricted master problem and the multiple improving directions.

On 10 instances involving 2,000 constraints and up to 10,000 variables for *simultaneous vehicle and crew scheduling problems in urban mass transit systems* (VCS), IPS reduces CPU times by a factor of 3.53 compared to CPLEX's PS (Elhallaoui et al. 2011; Raymond et al. 2010b). These set partitioning problems have degeneracy levels of about 50 %. Although these instances have been randomly generated, Haase et al. (2001) have constructed their generator such that *the main features of real-life VCS are reflected*.

IPS is also tested on 14 instances of *aircraft fleet assignment* (FA). These consist in maximizing the profits of assigning a type of aircraft to each flight segment over a horizon of one week. The content of the multi-commodity flow formulation for each of these instances can be resumed with these ballpark figures: a degeneracy level of 65 %, 5,000 constraints and 25,000 variables. IPS reduces CPU times by a factor of 12.30 on average compared to CPLEX's PS. While both types of problems are solved by column generation, the IPS methodology is tested by saving thousands of variables from the generator obviously including the optimal ones.

In these fleet assignment problems, an upper bound of 1 can explicitly be imposed on arc flow variables (see the discussion in Sect. 4.3). Hence, degeneracy occurs for basic variables at 0 or at 1. The comparison is still done against CPLEX's PS but the upper bounds are explicitly added in both solvers. CPU times are reduced by a factor of 20.23 on average for these LPs Raymond et al. (2009). These IPS algorithms have yet to be compared together.

On another note, opposing the convex combination to its column components content has been tested as follows: Computational experiments conducted with a hybrid algorithm starting with the classical generated columns (the surrogate variables) for the restricted master problem and ending with the column components (the original $\mathbf{x}$-variables) for the compact formulation shows improving average factors of 3.32 and 13.16 compared to CPLEX's PS on the previously mentioned VCS and FA problems Metrane et al. (2010).

5 Designing around compatibility

As supported by the vector subspace and the subspace basis flexibility, the compatibility notion is indeed quite flexible. In fact, when solving particular linear programs, the existing specialized algorithms, devised within the confines of IPS, that have proved to be successful share the common trait of being designed around and for compatibility. Sections 5.1 and 5.2, respectively, address network and set partitioning problems.

5.1 Network flow

In the context of the capacitated minimum cost flow problem, one refers to a solution $\mathbf{x}$ as a cycle free solution if the network contains no cycle composed only of free arcs, see Ahuja et al. (1993). Any such solution can be represented as a collection of free arcs (the nondegenerate basic arcs forming a forest) and all other arcs at their lower or upper bounds. The column vectors of the free arcs form $\mathbf{A}_F$, see Fig. 4.

According to Proposition 3 and the flow conservation equations, an arc at its lower or upper bound is compatible if and only if it can be written in terms of the unique subset of free arcs forming a cycle with it Desrosiers et al. (2014). Therefore, a compatible arc simply links two nodes belonging to the same tree of the forest. By opposition, an incompatible arc links two nodes of two different trees.

This characterization allows us to better understand the mechanism of improving cycles in networks. A feasible solution is optimal if and only if there is no negative cost directed cycle on the residual network. Two types of cycles can result from the pricing problem: a cycle containing a single compatible arc together with some free arcs of the same tree, or a cycle containing at least two incompatible arcs together with possibly some free arcs from different trees of the forest.

In Fig. 5, the dotted arc $(8, 9)$ is compatible and forms a directed cycle with the free arcs $(9, 10)$, $(10, 11)$, and $(11, 8)$. Indeed, the associated column in rows 8 through 11 are such that

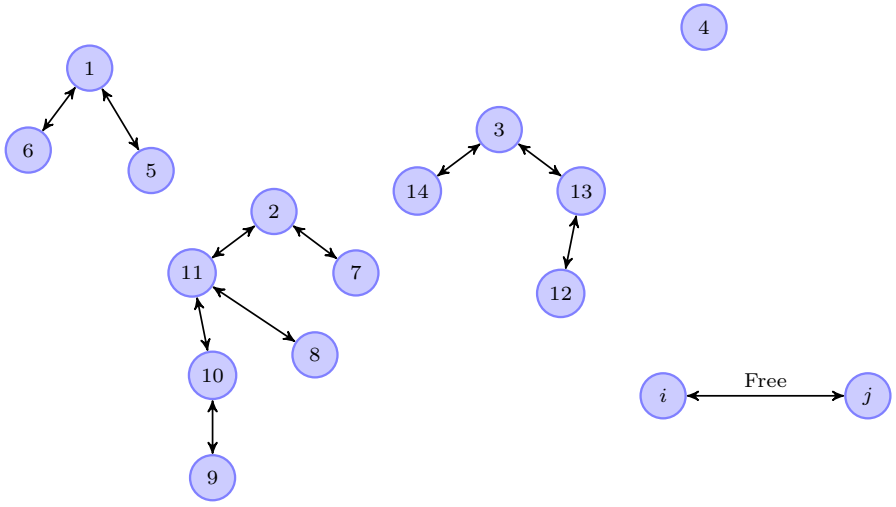


Fig. 4 Forest of free arcs in $\mathbf{A}_F$ on a residual network

$$\begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix} = - \begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \end{bmatrix} - \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

The dashed arc $(6, 9)$ links two different trees and is therefore incompatible. This is also the case for some other arcs, e.g., $(8, 12)$, $(3, 4)$ and $(4, 6)$. The reader may verify that the sum of the associated four columns is compatible as it can be written as the negated sum of the columns associated with the free arcs $(9, 10)$, $(10, 11)$, $(11, 8)$ and $(12, 13)$, $(13, 3)$. Indeed, these nine arcs form a directed cycle in the residual network.

Since IPS only makes nondegenerate pivots, it converges to optimality in a finite number of iterations on integral data network flow problems. Desrosiers et al. (2013) show that IPS is strongly polynomial for binary network problems, e.g., assignment, single shortest path, and unit capacity maximum flow. With a slight modification, it becomes strongly polynomial for solving the capacitated minimum cost network flow problem. The proposed contraction-expansion IPS-based algorithm is similar to the minimum mean cycle-canceling algorithm, see Goldberg and Tarjan (1989); Radzik and Goldberg (1994). On a network comprising n nodes and m arcs, it performs $O(m^2 n)$ iterations and runs in $O(m^3 n^2)$ time for arbitrary real-valued arc costs.

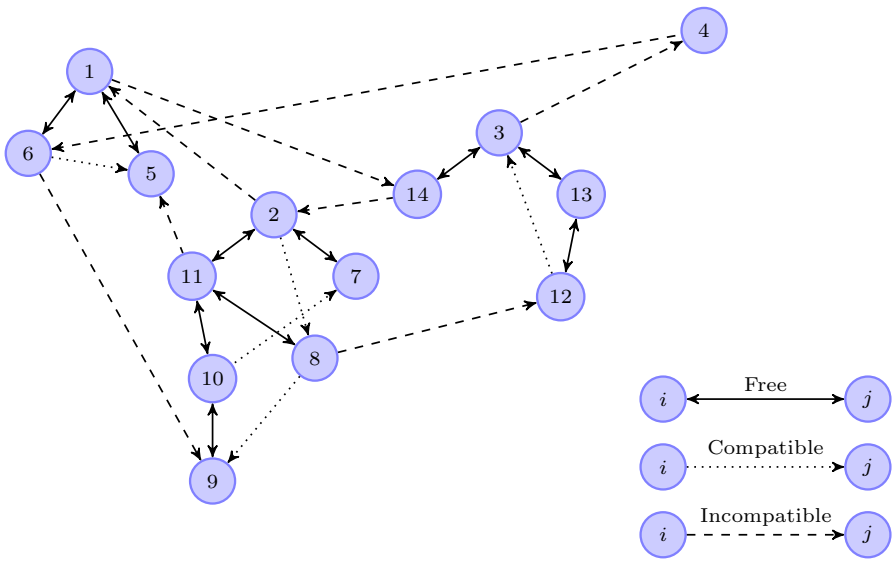


Fig. 5 Compatibility characterization of degenerate arcs on a residual network

5.2 Set partitioning

The set partitioning problem (SPP) can be formulated as the binary linear program

$$\min \quad \mathbf{c}^\intercal \mathbf{x} \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{1}, \quad \mathbf{x} \in \mathbb{B}^n, \tag{41}$$

where $\mathbf{A} \in \mathbb{B}^{m \times n}$. This formulation can be seen as a generic model encountered in various applications, namely, in vehicle routing and crew scheduling, and many more where the aim is to perform a clustering of the rows. In such applications, each set partitioning constraint can be associated with a task $i \in \{1, \ldots, m\}$ that must be *covered* exactly once. Such a formulation arises naturally from applying a Dantzig–Wolfe reformulation to a multi-commodity flow model in which each vehicle or crew member is represented by a separate commodity, see Desrosiers et al. (1995) and Desaulniers et al. (1998).

In order to express the fundamental exchange mechanism of set partitioning solutions, we assume that the current vector $\mathbf{x}_F$ is binary. Figure 6 should help demystify the concept of compatibility on SPP.

In the left-hand side, we find the binary input solution defined by the three independent columns of $\mathbf{A}_F$. According to Proposition 3, the next column identified by x_4 is compatible with the given partition as it covers exactly the first and third clusters. The third set shows the two incompatible columns x_5 and x_6. None can be generated by the columns of $\mathbf{A}_F$. However, their addition is compatible with the given partition as it covers the first and second clusters of rows. Finally, the right-hand side set exhibits three incompatible columns, x_7, x_8 and x_9: their combination with equal weights of 1/2 is compatible as it covers the second and third clusters of the row partition. Notice that this combination breaks the integrality of the next solution.

The compatible columns are readily available as the spanning set of $\mathbf{A}_F$ as per Proposition 3: a binary column is compatible if and only if it covers some of the clusters. Therefore, the interpretation of compatibility can be seen as a question of coverage. When the selected column is a combination of incompatible columns, the exchange mechanism removes elements from some clusters to insert them back in other clusters.

When the input solution is fractional, the mathematical definition of compatibility (Definition 1 or Proposition 3) still holds but the interpretation loses practical meaning. In order to sidestep this unfortunate loss, we can fall back on $\mathbf{\Lambda}_r$ (Definition 2) and adjust the subspace basis interpretation with respect to an aggregation/clustering scheme. The idea is to assume that certain tasks are done

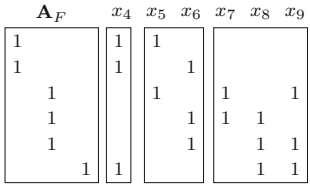


Fig. 6 Compatibility characterization for set partitioning binary solution $\mathbf{x}_F$

together and it is the cornerstone of DCA as described in the following section. Historically speaking, DCA is a self-standing algorithm devised for set partitioning models which provides an easy way to define a specialized vector subspace that *often* shares the properties of the one designed for IPS. The next section discusses the differences and similarities that arise between the two methods. We insist that it is in retrospective that the ties between DCA and IPS have been better understood.

6 Dynamic constraint aggregation

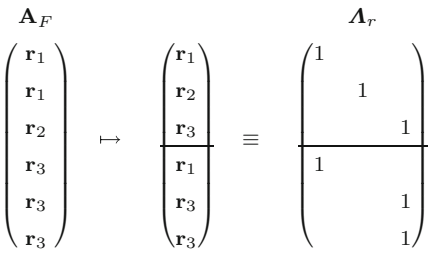
It is the first time DCA and IPS are studied in parallel. While they share several similarities, we hope to dissolve the confusion that arises between the two theories by highlighting their differences. IPS relies on the linear algebra framework to ascertain its faith and is therefore constructive by nature. It turns out that DCA is also born from a constructive design. This design is however limited by the embryonic intuition of a *reduced basis*. Let it be said that DCA is an intuitive precursor to IPS.

In a nutshell, the differences spring forth from the choice of the vector subspace to represent the current solution. Recall the subspace basis $\mathbf{\Lambda}_f$ and the equivalent generic transformation $\mathbf{T}^{-1}$, DCA disregards this choice and uses the general subspace basis format. It constructs $\mathbf{\Lambda}_r, r \geq f$, large enough to span $\mathbf{A}_F$. Let us see how and why it performs well.

In Sect. 6.1, we derive a row partition using a simple construction. The method is then applied in Section 6.2 on a set partitioning problem. The *inexistent* pricing step of DCA is explained in Sect. 6.3. An overview of the algorithm is illustrated in Sect. 6.4. Section 6.5 meditates on the integrality dimension of SPP. Finally, computational results are summarized in Sect. 6.6.

6.1 $\mathbf{\Lambda}_r$ derived from the identical rows of $\mathbf{A}_F$

The idea behind DCA is similar to the first step of a LU-decomposition for $\mathbf{A}_F$. Some of the rows which can easily be identified as null entries after elimination are actually identical rows. Of course, such a strategy might propose a set of constraints where some rows are redundant because linear independence is not thoroughly verified. Nevertheless, the separation between unique and identical rows induces a partition of the rows. The size of the partition is expressed as *the number of clusters* $r \geq f$ *of identical rows in* $\mathbf{A}_F$. Consider the following generic example where $\mathbf{A}_F$ contains six rows distributed into three clusters. The first step $(\mapsto)$ consists of a permutation of the lines such that the top rows are unique and the bottom rows are duplicates. The second step $(\equiv)$ provides a subspace basis $\mathbf{\Lambda}_r$ where each row cluster is associated with a unique 1-column identifier. Observe that by construction the top rows always correspond to I_r in the vector subspace, hence the subspace basis is of the form $\mathbf{\Lambda}_r = \begin{bmatrix} \mathbf{I}_r \\ \mathbf{M} \end{bmatrix}$:



The subspace basis $\boldsymbol{\Lambda}_r$ may over-span $\mathbf{A}_F$ if $r > f$. In other words, when $r = f$ we get from Proposition 3 that the decomposition is minimal and exactly corresponds to a generic basis of IPS. When $r > f$, $\boldsymbol{\Lambda}_r$ may lead to degenerate pivots although hopefully less than with PS.

6.2 DCA on set partitioning models

DCA is devised solely for the set partitioning problem. It capitalizes on the compatibility interpretation and characterization of set partitioning optimal solutions. A binary solution to (41) is usually highly degenerate. Indeed, in typical vehicle routing and crew scheduling applications, a cluster covers several tasks, say on average $\bar{m}$, which implies that the number of variables assuming value one in the basis is of the order $\mathrm{m}/\bar{m}$. The idea of row *aggregation* is born.

Assume for the moment that the linear relaxation of the set partitioning formulation (41) is written in standard form, that is,

$$z^\star := \min \quad \mathbf{c}^\intercal \mathbf{x} \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{1}, \quad \mathbf{x} \geq \mathbf{0}. \tag{42}$$

We present three situations that can occur in DCA. The first assumes the current solution is binary while the second and third consider a fractional input for which the partition is the same as the IPS decomposition for the former and different for the latter.

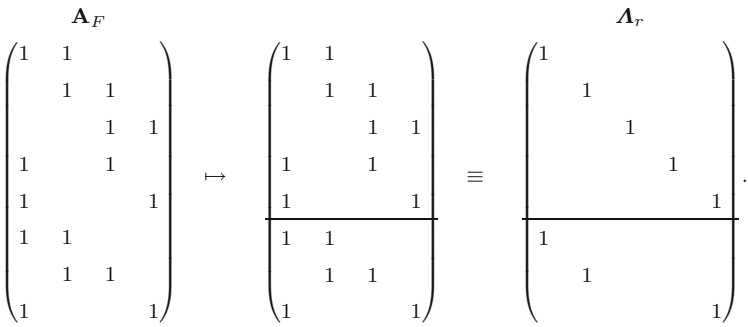
If $\mathbf{x}_F$ is binary, the corresponding columns of $\mathbf{A}_F$ are disjoint and induce a partition of the row set into f clusters. From $\mathbf{A}_F$, it is easy to construct a reduced working basis: take a single row from each cluster and therefore, upon a permutation of the columns and rows of $\mathbf{A}$, matrix $\mathbf{A}_{PF}$ is $\mathbf{I}_f$. This is illustrated with the following integer solution: $(x_1, x_2, x_3) = (1, 1, 1)$:

$$\mathbf{A}_F \quad \boldsymbol{\Lambda}_f$$
$$\begin{pmatrix} 1 & & \\ 1 & & \\ 1 & & \\ & 1 & \\ & 1 & \\ & & 1 \\ & & 1 \end{pmatrix} \mapsto \left(\begin{array}{ccc} 1 & & \\ & 1 & \\ & & 1 \\ \hline 1 & & \\ 1 & & \\ & 1 & \\ & & 1 \end{array}\right) \equiv \left(\begin{array}{ccc} 1 & & \\ & 1 & \\ & & 1 \\ \hline 1 & & \\ 1 & & \\ & 1 & \\ & & 1 \end{array}\right).$$

If $\mathbf{x}_F$ is fractional, the row partition is again derived from the number of clusters $r \geq f$ of identical rows of $\mathbf{A}_F$. If $r = f$, we can again construct a working basis as in IPS. Take the first row from each cluster to form $\mathbf{A}_{PF}$ while the $m - f$ rows of $\mathbf{A}_{ZF}$ are copies of the f independent rows of $\mathbf{A}_{PF}$. Right multiplying $\mathbf{A}_F$ by $\mathbf{A}_{PF}^{-1}$ provides the subspace basis $\boldsymbol{\Lambda}_f = \begin{bmatrix} \mathbf{I}_f \\ \mathbf{A}_{ZF}\mathbf{A}_{PF}^{-1} \end{bmatrix}$. This alternative subspace basis is similar to the one obtained from a binary solution. This is illustrated with the following 3-variable fractional solution $(x_1, x_2, x_3) = (0.5, 0.5, 0.5)$:

$$\mathbf{A}_F \quad \boldsymbol{\Lambda}_f$$
$$\begin{pmatrix} & 1 & 1 \\ & 1 & 1 \\ & 1 & 1 \\ 1 & & 1 \\ 1 & & 1 \\ 1 & 1 & \\ 1 & 1 & \\ 1 & 1 & \end{pmatrix} \mapsto \left(\begin{array}{ccc} & 1 & 1 \\ 1 & & 1 \\ 1 & 1 & \\ \hline & 1 & 1 \\ & 1 & 1 \\ 1 & & 1 \\ 1 & 1 & \\ 1 & 1 & \end{array}\right) \equiv \left(\begin{array}{ccc} 1 & & \\ & 1 & \\ & & 1 \\ \hline 1 & & \\ 1 & & \\ & 1 & \\ & & 1 \\ & & 1 \end{array}\right).$$

The third example shows a subspace basis $\boldsymbol{\Lambda}_r$ with $r > f$ induced by $\mathbf{x}_F = (x_1, x_2, x_3, x_4) = (0.5, 0.5, 0.5, 0.5)$. $\mathbf{A}_F$ comprises five clusters of identical rows, hence $\boldsymbol{\Lambda}_r$ has a dimension of $r = 5$. The row vectors satisfy $\mathbf{r}_1 + \mathbf{r}_3 = \mathbf{r}_2 + \mathbf{r}_5$ and IPS would have discarded one of these to construct $\mathbf{A}_{PF}$ of dimension $f = 4$.



The idea of compatibility steered the research and the implementation of DCA and its variants. In the context of routing and scheduling, *compatibility* thus means generating itineraries or schedules which precisely respect the current clustering into multi-task activities. Of course, this is in perfect harmony with Definition 2. From the above discussion, we see that the subspace basis $\mathbf{\Lambda}_r$ is derived from the solution of the linear relaxation formulation (42). However, the process can be initialized from any partition $\mathbf{\Lambda}_r$ of the rows: this can be done in a heuristic manner, even infeasible. This simply results in a linear transformation $\mathbf{T}^{-1}$ applied on the system of equality constraints, updated when needed.

6.3 Resolution process

The resolution process of DCA uses several properties presented in Sect. 4. In fact, RMP_{FC} is implemented under the name of *aggregated restricted master problem* by Elhallaoui et al. (2005). The missing part of the puzzle, now provided by IPS, is the pricing problem (38). Since DCA did not have such a feature when it was originally designed, let us take a look how it manages to pursue the optimal solution.

Let us go back to basics and consider an optimal basic solution to RMP_{FC}, that is, the formulation (37) restricted to the column set $F \cup C$ (the compatible variables). From now on assume F represents the index set of the free variables in this so-called current solution, where $f \leq r$, and $\boldsymbol{\psi}_R$ is an optimal dual vector. If $f = r$, the matrix $\mathbf{A}_{RF}$ is the current working basis as in IPS; otherwise $f < r$ and the working basis is the final PS basis provided with the optimal solution, the one that serves to compute $\boldsymbol{\psi}_R$. In any case, we have $\bar{\mathbf{c}}_F = \mathbf{c}_F - \boldsymbol{\psi}_R^\intercal \mathbf{A}_{RF} = \mathbf{0}$ for basic variables $\mathbf{x}_F$ and $\bar{\mathbf{c}}_C = \mathbf{c}_C - \boldsymbol{\psi}_R^\intercal \mathbf{A}_{RC} \geq \mathbf{0}$ by optimality. The current solution is optimal if

$$\bar{\mathbf{c}}_I = \mathbf{c}_I - \boldsymbol{\psi}_R^\intercal \mathbf{A}_{RI} - \boldsymbol{\psi}_Z^\intercal \bar{\mathbf{A}}_{ZI} \geq \mathbf{0}.$$

While solving RMP_{FC}, the neglected constraints in the row set Z have no dual information on $\boldsymbol{\psi}_Z$. As in IPS, reduced costs $\bar{\mathbf{c}}_I$ are partial to the chosen partition. Yet, optimality of current solution is either true or false. The same reduced costs written with respect to the original dual vector, $\bar{\mathbf{c}}_I = \mathbf{c}_I - \boldsymbol{\pi}^\intercal \mathbf{A}_I$, highlight the

possibility of *adapting* dual vector $\boldsymbol{\psi}_R$ to $\boldsymbol{\pi}$. The answer appears in (40), $[\boldsymbol{\pi}_R^\intercal, \boldsymbol{\pi}_Z^\intercal] = [\boldsymbol{\psi}_R^\intercal - \boldsymbol{\psi}_Z^\intercal \mathbf{M}, \boldsymbol{\psi}_Z^\intercal]$, which leads us to

$$\boldsymbol{\pi}_R^\intercal + \boldsymbol{\pi}_Z^\intercal \mathbf{M} = \boldsymbol{\psi}_R^\intercal. \tag{43}$$

As every column of the binary matrix $\mathbf{M}$ in a set partitioning problem identifies the remaining rows of a cluster, it means that $\psi_i, i \in R$, the dual variable of a cluster, must be distributed across the rows of its cluster, that is, (43) reads as $\sum_{\ell \in R_i} \pi_\ell = \psi_i, \forall i \in R$. Note that $\forall i \in R$, no matter how the $\boldsymbol{\psi}_i$ are distributed over their respective clusters, $\bar{\mathbf{c}}_F = \mathbf{0}$ and $\bar{\mathbf{c}}_C \geq \mathbf{0}$ remain satisfied. Therefore, consider the following set of constraints:

$$\boldsymbol{\pi}^\intercal \mathbf{a}_j \leq c_j, \quad \forall j \in I, \tag{44}$$

$$\sum_{\ell \in R_i} \pi_\ell = \psi_i, \quad \forall i \in R. \tag{45}$$

Notice that the system (44)–(45) is about feasibility. On the one hand, it is indeed feasible which means that the existence of acceptable dual variables certifies the optimal status of the current solution. On the other hand, it is infeasible: some constraints from (44) are removed until one retrieves a vector $\boldsymbol{\pi}$. Given those $\boldsymbol{\pi}$ values, DCA next prices out variables as in PS. A small selection of negative reduced costs incompatible variables is presumptuously added to current $\mathbf{A}_F$, say columns $\mathbf{A}_{I'}, I' \subset I$, such that a new partition is induced by the identical rows in $[\mathbf{A}_F, \mathbf{A}_{I'}]$, where $f + |I'| \leq m$. This yields a new subspace basis $\boldsymbol{\Lambda}_r, r > f$, at which point the algorithm proceeds with a new iteration, solving RMP_{FC} over a new set of constraints and compatible variables.

In DCA, the exercise of distributing the dual multipliers is called *dual variable disaggregation*. Since the expectation of an optimal solution to (44)–(45) can be put on hold, the system can be preemptively constructed in such a way that the algorithm expects a new partition. In particular, Elhallaoui et al. (2010) use a low-rank incompatibility strategy. In this respect, the disaggregation is heuristic by design.

6.3.1 Column generation

The pricing step established in IPS is however now available for DCA. Furthermore, in the context of column generation, the columns of $\mathbf{A}$ being unknown and thus provided by a column generation pricing problem, the transformed matrix $\bar{\mathbf{A}}$ needs to be fabricated just the same. Indeed, the reader can verify that the improved pricing step would optimally solve by column generation (see Desrosiers et al. (2014)) the following problem, equivalent to (38) for a set partitioning problem written in standard form:

$$\begin{array}{lrcll} \max & \mu & & & \\ \text{s.t.} & \mu & \leq & c_j - \boldsymbol{\pi}^\intercal \mathbf{a}_j, & [y_j] \quad \forall j \in I, \\ & \sum_{\ell \in R_i} \pi_\ell & = & \psi_i, & \forall i \in R. \end{array} \tag{46}$$

In SPP formulations of vehicle routing problems, the column generation pricing problem is a constrained shortest path. Since $\bar{\mathbf{A}}$ carries the information about the chosen partition $\{R, S\}$, modifying the constrained shortest path generator to account for this assumed partition effectively lightens its computational burden. Coined Bi-dynamic constraint aggregation (BDCA), this strategy is accounted for in the second revision of DCA, see Elhallaoui et al. (2008). While the idea of transferring the partition information might seem intuitive, the content of the latter paper is hardly summarizable in a few lines. We therefore insist on the former idea rather than this one successful application.

6.4 DCA algorithm

Figure 7 presents the algorithm for DCA. Let us concentrate on the modifications brought to Steps 3–5. The dual variable disaggregation replaces the pricing problem in Step 3b. Since the latter no longer provides an improving direction, Step 4 is skipped altogether. In Step 5, we move on directly to the column partition update. The small selection of incompatible variables $I' \subseteq I$ accompanies the column set F in Step 1 to create a new row partition.

6.5 Maintaining integrality

In SPP, wishful thinking unites with practicality. Indeed, the binary structure of the technological constraints is highly prejudicial to identical appearances in $\mathbf{A}_F$ represented by positive variables. Recall the three proposed examples to support this claim. In fact, when the solution is binary, the master problem of DCA is the same as in IPS. Furthermore, the nature of the pricing problem is such that it identifies convex combination containing few number of variables. Researchers such as Zaghrouti et al. (2014) rapidly turned a keen eye on this feature in an effort to maintain the integrality throughout the resolution process. It is known as integral simplex using decomposition (ISUD).

It amounts to verifying that the solution of the pricing problem, finally recuperated from IPS, yields an improved binary solution, rejecting the associated direction otherwise. In this respect, the binary restriction is transferred to the

1 Generic basis $\mathbf{T}$, transformation $\mathbf{T}^{-1}$, row partition $\{R, \bar{R}\}$ of $\boldsymbol{\Lambda}_r$;
2 Compatibility with the row partition $\{R, \bar{R}\}$ of $\boldsymbol{\Lambda}_r$ <optional>;
3a Restricted master problem: solve RMP_{FC} to optimality <optional goto Step 5>;
3b Fetch a subset $I' \subseteq I$ from the dual variable disaggregation;
4 Skip the exchange mechanism;
5 Update the column partition $\{F, L, U\}$ and goto Step 1;

Fig. 7 DCA algorithmic steps

pricing problem, yet it is only used on a needed basis. Of course, the additional work imposed de facto on the pricing problem makes it more difficult to solve but if an optimal binary solution is obtained, it means the elimination of the branch-and-bound requirement. That is to say that when we aim to maintain integrality in the resolution process, it makes it hard not to endorse DCA's strategy. The latter exploits a fast partition scheme which works out exactly when the solution is binary and compatibility is easy to verify without the need of an inverse.

6.6 Computational results for DCA

For the aforementioned VCS problems with some 2,000 constraints together with average degeneracy levels between 40 and 50 %, the combination of these ideas within GENCOL, a column generation software system, allows a reduction of solution times by factors of 50 to 100 (4.86 DCA $\times$ 4.38 BDCA $\times$ 4.53 MPDCA). The improvement factors are compounded as the strategies mix well together. DCA is the original version which is improved upon when the partition information is exploited in the pricing problem for generating negative reduced cost columns. Firstly with the modified constrained shortest path problem (BDCA) and secondly with the use of a low-rank incompatibility strategy (MPDCA).

To overcome degeneracy encountered during the resolution of the restricted master problem, Benchimol et al. (2012) propose a *stabilized* DCA (SDCA) that incorporates the above mentioned DCA into the dual variable stabilization (DVS) method of Oukil et al. (2007). The rationale behind this combination is that DCA reduces the primal space whereas DVS acts in the dual space. Combining both thus allows to fight degeneracy from primal and dual perspectives simultaneously. This method is again designed for solving the linear relaxation of set partitioning type models only. The computational results obtained on randomly generated instances of the multi-depot vehicle scheduling problem show that the performance of SDCA is not affected by the level of degeneracy and that it can reduce the average computational time of the master problem by a factor of up to 7 with respect to DVS. While this is not a direct comparison with DCA, the reduction factor would be even greater. Indeed, many instances solved by DVS could not be solved by DCA alone.

While DCA is implemented in a column generation context, ISUD is still in the early phase and applies only to known columns. The latest work of Rosat et al. (2014) shows that the pricing problem can be modified with relatively simple cuts when directions are rejected. These cuts have a major impact on the mean optimality gap dropping to 0.21 % on some aircrew scheduling problems from 33.92 % in the first ISUD paper.

7 Positive edge

The identification of variables compatible with the row set R requires the computation of the transformed matrix $\bar{\mathbf{A}}^0_{ZN} = \mathbf{T}_Z^{-1}\mathbf{A}^0_{ZN}$, where $\mathbf{T}_Z^{-1} := [\,-\mathbf{M} \quad \mathbf{I}_{m-r}\,]$. For large-scale problems, this can be time consuming. To overcome this situation, Raymond et al. (2010a) propose the Positive Edge rule. The latter exploits a creative stochastic argument which in turn allows the use of matrix multiplication rules to reduce the computational penalty. PE allows to determine whether a variable y_j, $j \in N$, is compatible or not without explicitly computing vector $\bar{\mathbf{a}}_{Zj} = \mathbf{T}_Z^{-1}\mathbf{a}_j$. It is based on the observations put forth in Sect. 7.1. Its statement is unveiled in Sect. 7.2 which also discusses its scope. Computational results are made available in Sect. 7.3.

7.1 Observations

Recall that if $\mathbf{a}_j$ is compatible, then $\bar{\mathbf{a}}_{Zj} = \mathbf{0}$. Hence, for any vector $\mathbf{v} \in \mathbb{R}^{m-r}$, we must have $\mathbf{v}^\intercal \bar{\mathbf{a}}_{Zj} = 0$. Otherwise, $\bar{\mathbf{a}}_{Zj} \neq \mathbf{0}$ and

$$\mathbf{v}^\intercal \bar{\mathbf{a}}_{Zj} = 0 \text{ if and only if } \mathbf{v} \perp \bar{\mathbf{a}}_{Zj}, \tag{47}$$

that is, if and only if $\mathbf{v}$ and $\bar{\mathbf{a}}_{Zj}$ are orthogonal. Intuitively, this has a probability of zero for a continuous random vector $\mathbf{v}$. Define $\mathbf{w}^\intercal := \mathbf{v}^\intercal \mathbf{T}_Z^{-1}$. Then, for any variable y_j, $j \in N$,

$$\mathbf{v}^\intercal \bar{\mathbf{a}}_{Zj} = \mathbf{v}^\intercal \mathbf{T}_Z^{-1}\mathbf{a}_j = \mathbf{w}^\intercal \mathbf{a}_j. \tag{48}$$

The expression (48) is similar to $\mathbf{c}_B^\top \bar{\mathbf{a}}_j = \boldsymbol{\pi}^\top \mathbf{a}_j$ in the computation of the reduced cost of variable x_j, where $\boldsymbol{\pi}$ is the vector of dual variables associated with constraint set $\mathbf{Ax} = \mathbf{b}$.

Computer architecture obliges, the continuous support is traded for a discrete one thus rendering the orthogonal probability to a nonzero value, although shown to be very small by Raymond et al. (2010a). We skip the proof and present only the random vector construction whose elements answer to the following definition.

Definition 3 Within the specifications of a computer with 32-bit words, a nonzero rational number $\mathcal{F} \in \mathbb{Q}_0$ with a discrete distribution SEM_{32} is a single precision floating-point number where the sign bit S, exponent E, and mantissa M are independent and follow the discrete uniform distributions $S \sim U[0,1]$, $E \sim U[64,191]$, and $M \sim U[0, 2^{23} - 1]$.

The random distribution SEM_{32} is symmetric around a zero mean ($\mu_{\mathcal{F}} = 0$) with a huge dispersion, its standard deviation being $\sigma_{\mathcal{F}} > 2^{60}$Towhidi et al. (2014). The random vector $\mathbf{v} \in \mathbb{Q}_0^{m-r}$ is such that all $m - r$ components are independent and identically distributed SEM_{32}.

7.2 PE rule

Within the scope of IPS, PE is a compatibility test which identifies nondegenerate improving pivots. It is a statistical test whereby the null hypothesis assumes the vector $\mathbf{a}$ is incompatible until sufficient evidence is provided to conclude otherwise. On a more abstract level, PE is a membership test. In a linear algebra framework, PE indeed amounts to stochastically testing whether a given vector belongs to a subspace. Two types of error may surface, the vector $\mathbf{a}$ is assumed compatible but it is not or the vector is assumed incompatible when it is.

7.2.1 Positive edge rule

Let $\mathbf{v} \in \mathbb{Q}_0^{m-r}$ be a random SEM_{32} vector. A vector $\mathbf{a} \in \mathbb{R}^m$ is *considered compatible with vector subspace* $\mathbf{V}(\mathbf{\Lambda}_r)$ if $\mathbf{w}^\intercal \mathbf{a}_j = \mathbf{0}$.

Since the operation amounts to a dot product, this means that a compatible variable is recognized in $O(m)$ time using the original vector $\mathbf{a}_j$. Researchers suggested to first consider compatible variables. Indeed, since these variables exist in the original formulation, their identification can benefit both the pricing step in IPS as well as provide additional information for pivot-selection rules in PS. As such, we believe the foremost purpose of PE is the *identification* of compatible variables.

With respect to IPS, the pricing problem over compatible variables reduces to $\min_{j \in C} \tilde{d}_j$ while that over the incompatible ones is identical to (13) [or (25) for linear programs in standard form] but with $\mathbf{y}_N$ replaced by $\mathbf{y}_I$. Notice that considering the set C is solely from a theoretical point of view, while in practice the compatibility test is only performed for variables with negative reduced costs or a subset of them.

PE has also been tested independently of IPS, that is, by selecting entering variables in PS among the compatible set in priority. We provide details in the computational results below.

7.3 Computational results for PE

The proof of concept is provided in Raymond et al. (2010a) by using two external procedures within `CPLEX`'s black box environment. A direct implementation in `COIN-OR`'s CLP, where it has been combined with the Devex pricing criterion, is presented in Towhidi et al. (2014). The proposed implementation uses a two-dimensional rule: for a variable x_j, $j \in N$, the first dimension computes the reduced cost $\bar{c}_j = c_j - \boldsymbol{\pi}^\intercal \mathbf{a}_j$, whereas the second evaluates $\mathbf{w}^\intercal \mathbf{a}_j$. PE identifies $C_\mathbf{w} = \{j \in N | \mathbf{w}^\intercal \mathbf{a}_j = 0\}$ and $I_\mathbf{w} = \{j \in N | \mathbf{w}^\intercal \mathbf{a}_j \neq 0\}$. Let $\bar{c}_{j_\star}$, $j_\star \in C_\mathbf{w} \cup I_\mathbf{w}$, be the smallest reduced cost and $\bar{c}_{j_\mathbf{w}}$, $j_\mathbf{w} \in C_\mathbf{w}$, be the smallest one for a compatible variable. The current solution is optimal if $\bar{c}_{j_\star} \geq 0$. Compatible variables are preferred to enter the basis except if $\bar{c}_{j_\star}$ is much smaller than $\bar{c}_{j_\mathbf{w}}$. Given a parameter $0 \leq \alpha < 1$, the selection rule is:

$$\text{if } c_{j_\star} < 0 \text{ and } c_{j_w} < \alpha c_{j_\star}, \text{ then select } x_{j_w} \text{ else } x_{j_\star}. \tag{49}$$

Tested with $\alpha = 0.5$ on 32 problems from Mittelmann's library Koch et al. (2011) which contains instances with a wide range of degeneracy levels, computational results show that below a degeneracy level of 25 %, PE is on average neutral while above this threshold, it reaches an average runtime speedup of 2.72, with a maximum of 4.23 on an instance with a 75 % degeneracy level.

8 Conclusions

This paper presents a survey of three recent tools for dealing with primal degeneracy in linear programming. While DCA appears first in the context of set partitioning models encountered in many routing and scheduling formulations solved by branch-and-price, IPS extends the concept to linear programs in general. Both methods partition the set of constraints in two parts, at every iteration, based on the values taken by the basic variables. This can be seen as a dynamic application of the Dantzig–Wolfe decomposition principle.

More specifically in IPS, one part of the constraints appears in the pricing problem as a homogeneous linear system (together with nonnegative variables) while the other part (together with the bound intervals on the variables) is used in the master problem to complete the exchange mechanism from one feasible solution to the next. PE adds a compatibility test layer, done in polynomial time, to the traditional reduced cost pricing of nonbasic variables. That is, it identifies those entering variables that belong to the current vector subspace and are likely to lead to nondegenerate pivots, if any. Otherwise, the IPS pricing step identifies a convex combination of incompatible ones which also ultimately leads to a nondegenerate pivot until optimality is reached in a finite number of iterations. Computational results reported from the literature show a large reduction on CPU times attributed to the diminution of degenerate pivots.

This paper also unifies IPS and DCA through a new interpretation in terms of the usage of two different subspace bases spanning the columns of the master problem. On the one hand, the subspace basis of IPS is made of the column vectors associated with the nondegenerate (or free) basic variables. On the other hand, that in DCA is derived from a partition of the rows into clusters, such as the one observed in any integer solution. This subspace basis has the fundamental property that it at least spans the free variable vectors. Therefore, the dimension of the subspace basis in DCA may be sometimes larger rather than equal to the number of free variables and this is the reason why some degenerate pivots may occur. As such, while every iteration of IPS is nondegenerate, DCA may encounter some degenerate pivots.

What does the future look like? While the theory behind IPS is sound and relatively straightforward, a general implementation is certainly a major concern. It is however hard to discard the specific structures of different families of LP problems. In this respect, the reader can think of several specializations of IPS to well structured problems such as network flows, multi-commodity flows, and linear relaxations of set partitioning and set covering problems.

The improved pricing step is the bottleneck of the method and needs to be handled with a great deal of insight. An efficient all-purpose implementation requires a significant amount of work and forces us to think about the different acceleration strategies that were presented herein. To name but a few, we have the partial pricing, the flexible subspace basis, the Dantzig–Wolfe surrogate variable environment and of course the infamous compatibility concept. The question that remains to be answered is whether trigger points for the usage of these ideas can be automated.

We are also looking at an implementation of these ideas within column generation, its adaptation to the dual simplex algorithm and to convex optimization, and its impact on the right-hand side sensitivity analysis, indeed the interpretation of the dual variables in the context of optimal degenerate solutions. Finally, the design of a completely efficient Integral Simplex algorithm for the set partitioning problem is a major goal.

Acknowledgments Jacques Desrosiers acknowledges the Natural Sciences and Engineering Research Council of Canada for its financial support.

References

Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: theory, algorithms, and applications. Upper Saddle River, New York

Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH (1998) Branch-and-price: column generation for solving huge integer programs. Oper Res 46(3):316–329. doi:10.1287/opre.46.3.316

Ben Amor HMT, Desrosiers J, Frangioni A (2009) On the choice of explicit stabilizing terms in column generation. Discret Appl Math 157(6):1167–1184. doi:10.1016/j.dam.2008.06.021

Benchimol P, Desaulniers G, Desrosiers J (2012) Stabilized dynamic constraint aggregation for solving set partitioning problems. Eur J Oper Res 223(2):360–371. doi:10.1016/j.ejor.2012.07.004

Bland RG (1977) New finite pivoting rules for the simplex method. Math Oper Res 2(2):103–107. doi:10.1287/moor.2.2.103

Valério de Carvalho JM (1999) Exact solution of bin-packing problems using column generation and branch-and-bound. Ann Oper Res 86:629–659. doi:10.1023/A:1018952112615

Valério de Carvalho JM (2002) LP models for bin-packing and cutting stock problems. Eur J Oper Res 141(2):253–273. doi:10.1016/S0377-2217(02)00124-8

Charnes A (1952) Optimality and degeneracy in linear programming. Econometrica 20(2):160–170. doi:10.2307/1907845

Dantzig GB (1963) Linear programming and extensions. Princeton University Press, Princeton

Dantzig GB, Wolfe P (1960) Decomposition principle for linear programs. Oper Res 8(1):101–111. doi:10.1287/opre.8.1.101

Desaulniers G, Desrosiers J, Ioachim I, Solomon MM, Soumis F, Villeneuve D (1998) A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In: Crainic T, Laporte G (eds) Fleet management and logistics. Springer, New York pp 57–93. doi:10.1007/978-1-4615-5755-5_3

Desrosiers J, Lübbecke ME (2011) Branch-price-and-cut algorithms. Wiley, New York. doi:10.1002/9780470400531.eorms0118

Desrosiers J, Dumas Y, Solomon MM, Soumis F (1995) Time constrained routing and scheduling. In: Ball M, Magnanti T, Monma C, Nemhauser G (eds) Handbooks in operations research and management science, Network routing, vol 8. Elsevier, New York, pp 35–139. doi:10.1016/S0927-0507(05)80106-9

Desrosiers J, Gauthier JB, Lübbecke ME (2013) A contraction-expansion algorithm for the capacitated minimum cost flow problem. Presentation at VeRoLog 2013, Southampton

Desrosiers J, Gauthier JB, Lübbecke ME (2014) Row-reduced column generation for degenerate master problems. Eur J Oper Res 236(2):453–460. doi:10.1016/j.ejor.2013.12.016
du Merle O, Villeneuve D, Desrosiers J, Hansen P (1999) Stabilized column generation. Discret Math 194:229–237. doi:10.1016/S0012-365X(98)00213-1
Elhallaoui I, Villeneuve D, Soumis F, Desaulniers G (2005) Dynamic aggregation of set partitioning constraints in column generation. Oper Res 53(4):632–645. doi:10.1287/opre.1050.0222
Elhallaoui I, Desaulniers G, Metrane A, Soumis F (2008) Bi-dynamic constraint aggregation and subproblem reduction. Comput Oper Res 35(5):1713–1724. doi:10.1016/j.cor.2006.10.007
Elhallaoui I, Metrane A, Soumis F, Desaulniers G (2010) Multi-phase dynamic constraint aggregation for set partitioning type problems. Math Progr 123(2):345–370. doi:10.1007/s10107-008-0254-5
Elhallaoui I, Metrane A, Desaulniers G, Soumis F (2011) An improved primal simplex algorithm for degenerate linear programs. INFORMS J Comput 23:569–577. doi:10.1287/ijoc.1100.0425
Fukuda K (1982) Oriented matroid programming. PhD thesis, University of Waterloo, Ontario, Canada
Gauthier JB, Desrosiers J, Lübbecke ME (2014) About the minimum mean cycle-canceling algorithm. Discret Appl Math. doi:10.1016/j.dam.2014.07.005
Goldberg AV, Tarjan RE (1989) Finding minimum-cost circulations by canceling negative cycles. J ACM 36(4):873–886. doi:10.1145/76359.76368
Haase K, Desaulniers G, Desrosiers J (2001) Simultaneous vehicle and crew scheduling in urban mass transit systems. Transp Sci 35(3):286–303. doi:10.1287/trsc.35.3.286.10153
Harris PMJ (1973) Pivot selection methods of the devex lp code. Math Progr 5(1):1–28. doi:10.1007/BF01580108
Karp RM (1978) A characterization of the minimum cycle mean in a digraph. Discret Math 23(3):309–311. doi:10.1016/0012-365X(78)90011-0
Klee V, Minty G (1972) How good is the simplex algorithm? In: Shisha O (ed) Inequalities, (Proc. Third Sympos., Univ. California, Los Angeles, Calif., 1969; dedicated to the memory ofTheodore S. Motzkin), vol 3. Academic Press, New York, pp 159–175
Koch T, Achterberg T, Andersen E, Bastert O, Berthold T, Bixby RE, Danna E, Gamrath G, Gleixner AM, Heinz S, Lodi A, Mittelmann H, Ralphs T, Salvagnin D, Steffy DE, Wolter K (2011) MIPLIB 2010. Math Progr Comput 3(2):103–163. doi:10.1007/s12532-011-0025-9
Löbel A (1998) Vehicle scheduling in public transit and Lagrangean pricing. Manag Sci 44(12):1637–1649. doi:10.1287/mnsc.44.12.1637
Lübbecke ME, Desrosiers J (2005) Selected topics in column generation. Oper Res 53(6):1007–1023. doi:10.1287/opre.1050.0234
Mamer JW, McBride RD (2000) A decomposition-based pricing procedure for large-scale linear programs: an application to the linear multicommodity flow problem. Manag Sci 46(5):693–709. doi:10.1287/mnsc.46.5.693.12042
Metrane A, Soumis F, Elhallaoui I (2010) Column generation decomposition with the degenerate constraints in the subproblem. Eur J Oper Res 207(1):37–44. doi:10.1016/j.ejor.2010.05.002
Omer J, Rosat S, Raymond V, Soumis F (2014) Improved primal simplex: a more general theoretical framework and an extended experimental analysis. Les Cahiers du GERAD G-2014-13, HEC Montréal, Canada
Oukil A, Ben Amor HMT (2007) Stabilized column generation for highly degenerate multiple-depot vehicle scheduling problems. Comput Oper Res 34(3):817–834. doi:10.1016/j.cor.2005.05.011
Pan PQ (1998) A basis deficiency-allowing variation of the simplex method for linear programming. Comput Math Appl 36(3):33–53. doi:10.1016/S0898-1221(98)00127-8
Perold AF (1980) A degeneracy exploiting LU factorization for the simplex method. Math Progr 19(1):239–254. doi:10.1007/BF01581646
Radzik T, Goldberg AV (1994) Tight bounds on the number of minimum-mean cycle cancellations and related results. Algorithmica 11(3):226–242. doi:10.1007/BF01240734
Raymond V, Soumis F, Metrane A (2009) Improved primal simplex version 3: cold start, generalization for bounded variable problems and a new implementation. Les Cahiers du GERAD G-2009-15, HEC Montréal, Canada
Raymond V, Soumis F, Metrane A, Desrosiers J (2010a) Positive edge: a pricing criterion for the identification of non-degenerate simplex pivots. Les Cahiers du GERAD G-2010-61, HEC Montréal, Canada
Raymond V, Soumis F, Orban D (2010b) A new version of the improved primal simplex for degenerate linear programs. Comput Oper Res 37(1):91–98. doi:10.1016/j.cor.2009.03.020

Rosat S, Elhallaoui I, Soumis F, Lodi A (2014) Integral simplex using decomposition with primal cuts. In: Gudmundsson J, Katajainen J (eds) Experimental algorithms, Lecture notes in computer science, vol 8504. Springer, New York, International Publishing, pp 22–33. doi:10.1007/978-3-319-07959-2_3

Ryan DM, Osborne MR (1988) On the solution of highly degenerate linear programmes. Math Progr 41(1–3):385–392. doi:10.1007/BF01580776

Sadykov R, Vanderbeck F (2013) Column generation for extended formulations. EURO J Comput Optim 1(1–2):81–115. doi:10.1007/s13675-013-0009-9

Schrijver A (1986) Theory of Linear and Integer Programming. Wiley, Chichester

Terlaky T, Zhang S (1993) Pivot rules for linear programming: a survey on recent theoretical developments. Annal Oper Res Anna OR 46–47(1):203–233. doi:10.1007/BF02096264

Towhidi M, Desrosiers J, Soumis F (2014) The positive edge criterion within COIN-OR's CLP. Comput Oper Res 49:41–46. doi:10.1016/j.cor.2014.03.020

Wolfe P (1963) A technique for resolving degeneracy in linear programming. J Soc Ind Appl Math 11(2):205–211. doi:10.1137/0111016

Zaghrouti A, Soumis F, El Hallaoui I (2014) Integral simplex using decomposition for the set partitioning problem. Oper Res 62(2):435–449. doi:10.1287/opre.2013.1247