http://pubsonline.informs.org/journal/opre/

OPERATIONS RESEARCH
Articles in Advance, pp. 1–14
ISSN 0030-364X (print), ISSN 1526-5463 (online)

Vector Space Decomposition for Solving Large-Scale Linear Programs

Jean Bertrand Gauthier,[a] Jacques Desrosiers,[a] Marco E. Lübbecke[b]

[a] HEC Montréal and GERAD, Montréal, Canada H3T 2A7; [b] Lehrstuhl für Operations Research, RWTH Aachen University, D-52072 Aachen, Germany

Contact: jean-bertrand.gauthier@hec.ca (JBG); jacques.desrosiers@hec.ca, http://orcid.org/0000-0003-0719-1500 (JD); marco.luebbecke@rwth-aachen.de, http://orcid.org/0000-0002-2635-0522 (MEL)

Received: September 8, 2016
Revised: March 20, 2017; September 28, 2017
Accepted: December 23, 2017
Published Online in Articles in Advance: August 2, 2018

Subject Classifications: linear programming; algorithms
Area of Review: Optimization

https://doi.org/10.1287/opre.2018.1728

Abstract. We develop an algorithmic framework for linear programming guided by dual optimality considerations. The solution process moves from one feasible solution to the next according to an exchange mechanism that is defined by a direction and a resulting step size. Part of the direction is obtained via a pricing problem devised in primal and dual forms. From the dual perspective, one maximizes the minimum reduced cost that can be achieved from splitting the set of dual variables in two subsets: one being fixed while the other is optimized. From the primal perspective, this amounts to selecting a nonnegative combination of variables entering the basis. The direction is uniquely complemented by identifying the affected basic variables, if any.

The framework is presented in a generic format motivated by and alluding to concepts from network flow problems. It specializes to a variety of algorithms, several of which are well known. The most prominent is the primal simplex algorithm where all dual variables are fixed: this results in the choice of a single entering variable commonly leading to degenerate pivots. At the other extreme, we find an algorithm for which all dual variables are optimized at every iteration. Somewhere in between these two extremes lies the improved primal simplex algorithm for which one fixes the dual variables associated with the nondegenerate basic variables and optimizes the remaining ones. The two last variants both bestow a pricing problem providing necessary and sufficient optimality conditions. As a result, directions yielding strictly positive step sizes at every iteration are also issued from these pricing steps. These directions move on the edges of the polyhedron for the latter while the former can also identify interior directions.

Funding: Jacques Desrosiers acknowledges the National Science and Engineering Research Council of Canada [Grant RGPIN/04401-2014] and the HEC Montréal Foundation for their financial support. Jean Bertrand Gauthier acknowledges the GERAD—Group for Research in Decision Analysis (of Montréal, Québec, Canada)—for its financial support.

Keywords: primal simplex algorithm • column generation • degeneracy • residual problem • optimized reduced costs • cycles • positive step size algorithms • vector space

1. Introduction

Degeneracy is a critical performance issue when solving linear programs with the simplex method in practice.

Dantzig and Thapa (2003, p. 167) suggest "that pivot-selection criteria should be designed to seek feasible solutions in directions away from degenerate and 'near'-degenerate basic feasible solutions, or better yet, driven by dual feasibility considerations." We revisit that statement with a different interpretation: when trying to avoid primal infeasible directions, one should consider pivot-selection (or pricing) rules that are also guided by *dual optimality*.

While Dantzig's pivot rule accurately measures the improvement rate of the objective function, the influence on the affected basic variables is taken for granted for every nonbasic variable unit change. When one realizes that not all affected basic variables *can* actually move, it becomes clear that the pricing rule suffers from a *visibility problem* in terms of basic variable space. Indeed, null step sizes can only happen when at least one basic variable is valued at one of its bounds. As such, degeneracy is a vicious cycle in the sense that solutions with higher levels of degeneracy are more likely to yield degenerate pivots. Additional computations based on the basis are incorporated in the pricing step of steepest edge and Devex pivot rules (Forrest and Goldfarb 1992, Harris 1973), which helps alleviate, but not eliminate, this shortcoming. We propose the elimination of basic variables leading to degeneracy from consideration in the pricing step: This is sufficient to overcome primal degeneracy. It grants more freedom in the dual variables thus allowing the pricing process flexibility in meeting dual optimality.

We propose a framework, which, given a basic feasible solution, *fixes the values of a subset of dual variables*

and optimizes the remaining ones for maximizing the minimum reduced cost. In the primal interpretation, this results in a pricing problem in which one selects a nonnegative combination of variables entering the basis. The way to divide the dual variables into two subsets relies on the choice of a vector subspace basis in the primal, capitalizing on the actual values taken by the basic variables. This opens a wide spectrum of possibilities. The general scheme resembles a dynamic decomposition like Dantzig and Wolfe's and inspires the name of *vector space decomposition*. It unifies a variety of specialized algorithms for linear and network programs. The most prominent special case is the primal simplex algorithm (PS) where *all* dual variables are fixed: this results in the choice of a single entering variable commonly leading to degenerate pivots. At the other extreme when *no* dual variables are fixed, that is, all dual variables are optimized at every iteration, we find the so-called minimum weighted cycle-canceling algorithm (MWCC) which is strongly polynomial when solving network flow problems (Goldberg and Tarjan 1989). Somewhere in between these two extremes lie the improved primal simplex algorithm (IPS) of Elhallaoui et al. (2011) for which one only fixes the dual variables associated with the nondegenerate basic variables, and the dynamic constraint aggregation method (DCA) of Elhallaoui et al. (2005) specifically designed to overcome degeneracy in the context of set partitioning models when solving the linear relaxation by column generation (see Lübbecke and Desrosiers 2005). Building our unified framework is motivated and enabled by network flow analogies seen in Gauthier et al. (2017). Alluding to Dantzig and Thapa (1997), our proposal can also be interpreted as a very general *dual guided* pivot rule.

Besides having a common theoretical container for such seemingly different algorithms as PS, IPS, DCA, MWCC, and (yet unknown) others, we feel that the framework itself opens a new avenue of generally coping with degeneracy in simplex-like (and related) algorithms. Much more importantly, however, we see an utmost practical perspective for our work. The primal simplex method has strong competitors with the dual simplex and barrier methods (Bixby 2002). In particular in linear programming based branch-and-bound, primal algorithms are not the first choice. In column generation, however, the primal simplex algorithm is inherently encoded in the pricing mechanism, regardless of which algorithm is actually used to reoptimize the linear restricted master programs. Column generation gains more and more significance in solving well-structured very large-scale linear programs from practical applications. As a descendant of the primal simplex method, it inherits all difficulties with degeneracy, and the linear relaxations of combinatorial optimization problems particularly suffer from this. Our framework offers a general and flexible remedy, and yet, it allows (and benefits from) tailoring to the particular application at hand. Furthermore, it is fortunate that our framework plays particularly well with a number of alternative suggestions to cope with degeneracy or degeneracy-related effects in column generation, such as dual variable stabilization (du Merle et al. 1999, Ben Amor et al. 2009).

The paper is organized as follows. Section 2 takes a close look at the essential components of the framework. Several concepts that partake (or not) in the resolution process of a linear program are examined such as nondegenerate pivots, cycles and directions, and the so-called *residual problem*. Each of these is presented in a separate manner, whereas the last subsection ties everything together. Section 3 builds upon these ties and gives birth to the generic framework. Two propositions are exposed in Section 4, the first determines conditions guaranteeing positive step sizes at every iteration and the second shows that identified directions can be interior rather than along edges. We conclude in Section 5 with our contribution and some research perspectives.

2. Linear Program

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

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

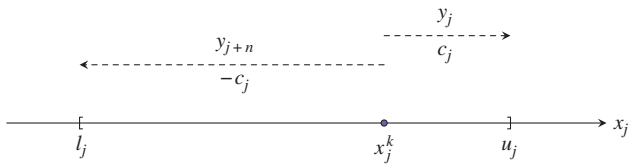
where $\boldsymbol{x}, \boldsymbol{c}, \boldsymbol{l}, \boldsymbol{u} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, and $m < n$. We assume that the matrix $\boldsymbol{A}$ is of full row rank, that LP (1) is feasible, and that z^* is finite. The vector of dual variables $\boldsymbol{\pi} \in \mathbb{R}^m$ associated with the equality constraints appears within brackets on the right-hand side.

2.1. Notation

Vectors and matrices are written in bold face by, respectively, using lower and upper case. We denote by $\boldsymbol{I}_r$ the $r \times r$ identity matrix and by $\mathbf{0}$ (respectively, $\mathbf{1}$) a vector/matrix with all zeros (respectively, ones) entries of contextually appropriate dimension. For an ordered subset $R \subseteq \{1,\dots,m\}$ of row indices and an ordered subset $P \subseteq \{1,\dots,n\}$ of column indices, we denote by $\boldsymbol{A}_{RP}$ the submatrix of $\boldsymbol{A}$ containing the rows and columns indexed by R and P, respectively. We further use standard linear programming notation like $\boldsymbol{A}_B\boldsymbol{x}_B$, the subset of basic columns of $\boldsymbol{A}$ indexed by B multiplied by the vector of basic variables $\boldsymbol{x}_B$. The index set of nonbasic columns N is used analogously.

In Section 2.2, we formulate the so-called *residual problem*, which allows the construction of an oracle generating feasible directions in Section 2.4. The latter also provides two alternative primal and dual conditions characterizing optimality for linear programs. Finally,

Figure 1. (Color online) Forward and Backward Variables for the Residual Problem



let us embark upon this generic framework in Section 2.5 by analyzing a linear transformation, the goal being to structure the technological constraints.

2.2. Residual Problem

It is a common concept in developing network flow algorithms to use a *residual network* to improve upon some intermediate solution by identifying augmenting flows; see Ahuja et al. (1993). Let us stay in the spirit of network flows and propose analogies for linear programs leading to primal-dual optimality conditions on the so-called residual problem.

We define the *residual problem* $LP(\boldsymbol{x}^k)$ with respect to a given solution $\boldsymbol{x}^k$ at iteration $k \geqslant 0$ as follows. Each variable x_j, $j \in \{1,\dots,n\}$, in the original LP (1) is replaced by two variables: the forward variable y_j of cost $d_j := c_j$ represents the possible increase $r_j^k := u_j - x_j^k$ of x_j relatively to x_j^k while the backward variable y_{j+n} of cost $d_{j+n} := -c_j$ represents its possible decrease $r_{j+n}^k := x_j^k - l_j$. In fact, the y-variables satisfy $y_j - y_{j+n} = x_j - x_j^k$, $\forall\, j \in \{1,\dots,n\}$. Moreover, only one from each pair can be used with a positive value, that is, the condition $y_j\, y_{j+n} = 0$ holds, $\forall\, j \in \{1,\dots,n\}$; see Figure 1.

Equivalent to LP (1), a formulation for the residual problem $LP(\boldsymbol{x}^k)$ is given as

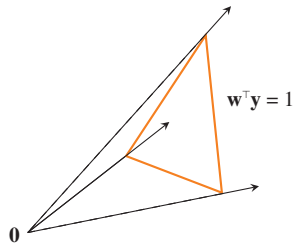
$$\begin{aligned} z^\star = z^k + \min\ & \boldsymbol{d}^\intercal \boldsymbol{y} \\ \text{s.t.}\ & \boldsymbol{K}\boldsymbol{y} = \boldsymbol{0} \quad [\pi] \\ & \boldsymbol{0} \leqslant \boldsymbol{y} \leqslant \boldsymbol{r}^k, \end{aligned} \tag{2}$$

where $z^k := \boldsymbol{c}^\intercal \boldsymbol{x}^k$, $\boldsymbol{d} := [d_j]_{j\in\{1,\dots,2n\}}$ is the cost vector, $\boldsymbol{y} := [y_j]_{j\in\{1,\dots,2n\}}$ contains the forward and backward variables, their residual capacities given by the upper bounds $\boldsymbol{r}^k := [r_j^k]_{j\in\{1,\dots,2n\}}$, and the matrix $\boldsymbol{K} := [\boldsymbol{A}, -\boldsymbol{A}] \equiv [\boldsymbol{k}_j]_{j\in\{1,\dots,2n\}}$ stands to remind us that the kernel (or null space) of this matrix is the set of solutions to $\boldsymbol{K}\boldsymbol{y} = \boldsymbol{0}$. A variable fixed to 0 is uninteresting, so $LP(\boldsymbol{x}^k)$ may be written using only the *residual variables*, that is, the y-variables with positive residual capacities within the set $J^k := \{j \in \{1,\dots,2n\} \mid r_j^k > 0\}$. While this might appear like a trivial statement, it is solely responsible for allowing the identification of directions inducing null step sizes as presented in the next section.

2.3. Directions and Cycles

By neglecting the upper bounds from (2), one obtains a cone whose every extreme ray induces an extreme direction originating from $\boldsymbol{x}^k$ as stated in Definition 1.

Figure 2. (Color online) The Cone $\{\boldsymbol{y} \geqslant \boldsymbol{0} \mid \boldsymbol{K}\boldsymbol{y} = \boldsymbol{0}\} \subseteq \mathbb{R}^{2n}_+$ Cut by $\boldsymbol{w}^\intercal \boldsymbol{y} = 1$



Definition 1 (Bazaraa et al. (1990)). Given an extreme ray $\boldsymbol{y} \in \{\boldsymbol{K}\boldsymbol{y} = \boldsymbol{0},\ \boldsymbol{y} \geqslant \boldsymbol{0}\} \subseteq \mathbb{R}^{2n}_+$, the components of *direction* $\vec{\boldsymbol{v}} \in \mathbb{R}^n$ are computed as differences:

$$\vec{v}_j = y_j - y_{j+n}, \quad \forall\, j \in \{1,\dots,n\}.$$

To find an improving direction, it then suffices to look within this set of extreme rays. This can be done via an optimization program. However, since optimizing in a cone is inconvenient with respect to any comparative measure (unless the optimal solution is the only extreme point), let us add the normalization constraint $\boldsymbol{w}^\intercal \boldsymbol{y} = 1$, where $\boldsymbol{w} > \boldsymbol{0}$ is a vector of arbitrary positive weights. This results in a cut cone where every nonnull extreme point corresponds to an extreme ray. In the same vein, by considering a different weight vector $\boldsymbol{w}$, the cutting plane would be slanted differently thus producing a modified set of extreme points, yet each of these would remain associated with the same extreme ray. The cone displayed in Figure 2 illustrates these extreme rays.

A *normalized direction* is a direction (abusively speaking by the correspondence between extreme rays and extreme points) for which $\boldsymbol{y}$ satisfies the intersection between the cone and the normalization constraint, that is,

$$\boldsymbol{y} \in \mathcal{N} := \{\boldsymbol{K}\boldsymbol{y} = \boldsymbol{0},\ \boldsymbol{y} \geqslant \boldsymbol{0},\ \boldsymbol{w}^\intercal \boldsymbol{y} = 1\} \subseteq \mathbb{R}^{2n}_+. \tag{3}$$

In the following, Definition 2 introduces the notion of *cycle*, which interchangeably characterizes a normalized direction, whereas Lemma 1 gives us a fundamental property regarding the cost and the reduced cost of such a cycle. Observe that the domain $\mathcal{N}$ does not depend on the current solution $\boldsymbol{x}^k$. However, by definition of the cone, a cycle may always be followed from $\boldsymbol{x}^k$ with a nonnegative step size $\rho \geqslant 0$. Definition 3 reconsiders the existence of the residual bounds $\boldsymbol{r}^k$ and characterizes the cycles that feature a strictly positive step size with respect to $\boldsymbol{x}^k$.

Definition 2. Let $\boldsymbol{w} \in \mathbb{R}^{2n}$ be a vector of strictly positive weights. A *cycle* W is the positive variable support of $\boldsymbol{y}$ in a normalized direction, that is, $W := \{j \in \{1,\dots,2n\} \mid y_j > 0, \boldsymbol{y} \in \mathcal{N}\}$.

The cost of a cycle W in (2) is computed as $d_W := \sum_{j\in W} d_j y_j$. A *negative cycle* is a cycle with negative cost. The reduced cost of variable x_j, $j \in \{1,\dots,n\}$, is defined as $\bar{c}_j := c_j - \boldsymbol{\pi}^\intercal \boldsymbol{a}_j$ while those of variables y_j and y_{j+n} are, respectively, $\bar{d}_j := c_j - \boldsymbol{\pi}^\intercal \boldsymbol{a}_j = \bar{c}_j$ and $\bar{d}_{j+n} := -c_j - \boldsymbol{\pi}^\intercal(-\boldsymbol{a}_j) = -\bar{c}_j$. The following provides the analogous result from networks that the cost and reduced cost of a cycle are equal.

Lemma 1. *The cost d_W and the reduced cost $\bar{d}_W$ of a cycle W are equal.*

Proof. A cycle W satisfies (3), hence $\sum_{j\in W} \boldsymbol{k}_j y_j = \boldsymbol{0}$ and $\bar{d}_W = \sum_{j\in W}(d_j - \boldsymbol{\pi}^\intercal \boldsymbol{k}_j) y_j = d_W - \boldsymbol{\pi}^\intercal \sum_{j\in W} \boldsymbol{k}_j y_j = d_W$. □

Definition 3. An *augmenting cycle* at $\boldsymbol{x}^k$ is a cycle W whose composing y-variables have positive residual capacities, that is, $r_j^k > 0$, $\forall\, j \in W$, or equivalently $W \subseteq J^k$.

When in the following we speak of negative cycles, we always refer to negative *augmenting* cycles as per Definition 3 and Lemma 1. Under the previous nomenclature, necessary and sufficient optimality conditions come together in a straightforward manner. In addition to the complementary slackness optimality conditions based on the reduced cost of the original x-variables, we repeat here two alternative conditions characterizing optimality for *LP* (1).

Proposition 1 (Gauthier et al. (2014, Theorem 4))**.** *A feasible solution $\boldsymbol{x}^k$ to LP (1) is optimal if and only if the following equivalent conditions are satisfied*:

Complementary slackness: *There exists a $\boldsymbol{\pi}$ such that*

$$\begin{array}{c} \bar{c}_j > 0 \;\Rightarrow\; x_j^k = l_j; \quad \bar{c}_j < 0 \;\Rightarrow\; x_j^k = u_j; \\ l_j < x_j^k < u_j \;\Rightarrow\; \bar{c}_j = 0. \end{array} \tag{4}$$

Primal: *$LP(\boldsymbol{x}^k)$ contains no negative cycle, that is,*

$$d_W \geqslant 0, \quad \text{for every cycle } W \text{ in } LP(\boldsymbol{x}^k). \tag{5}$$

Dual: *There exists a $\boldsymbol{\pi}$ such that the reduced cost of every residual variable of $LP(\boldsymbol{x}^k)$ is nonnegative, that is,*

$$\bar{d}_j \geqslant 0, \quad \forall\, j \in J^k. \tag{6}$$

2.4. Oracle

To prove the optimality of $\boldsymbol{x}^k$ or improve the current solution, we derive an oracle relying on the identification of cycles. One can be derived from the domain (3) and an objective function that effectively computes the cost (or the reduced cost) of each cycle properly as follows:

$$\min_{y\in\mathcal{N}} \boldsymbol{d}^\intercal \boldsymbol{y}. \tag{7}$$

Since upper bounds are removed from the domain $\mathcal{N}$, the oracle (7) may unfortunately identify a nonaugmenting cycle if any y-variable with zero residual capacity remains. Keeping track of the residual variables can be done by partitioning the x-variables according to their current values. To achieve this, let $\boldsymbol{x}^k$ be represented by $(\boldsymbol{x}_F^k, \boldsymbol{x}_L^k, \boldsymbol{x}_U^k)$, where the three subvectors refer to the set of free variables $F := \{j \in \{1,\dots,n\} \mid l_j < x_j^k < u_j\}$, at their lower bounds $L := \{j \in \{1,\dots,n\} \mid x_j^k = l_j\}$, and at their upper bounds $U := \{j \in \{1,\dots,n\} \mid x_j^k = u_j\}$, respectively. Let there be $f := |F|$ such free variables, $0 \leqslant f \leqslant n$. Observe that if $\boldsymbol{x}^k$ is basic then $0 \leqslant f \leqslant m$. Controlling the presence/absence of the residual variables while solving the oracle can then alternatively be achieved by imposing

$$y_j = 0, \quad \forall\, j \in U \quad \text{and} \quad y_{j+n} = 0, \quad \forall\, j \in L. \tag{8}$$

By the primal optimality conditions in Proposition 1, it is possible to improve intermediate solutions using negative cycles until an optimal solution is reached. In this respect, the step size ρ associated with the negative cycle W must satisfy $\rho\, y_j \leqslant r_j^k$, $\forall\, j \in W$, and this cycle is *canceled* when the step size is equal to $\rho := \min_{j\in W} r_j^k / y_j > 0$.

2.4.1. Primal Simplex Algorithm. Consider a basic solution $\boldsymbol{x}^k$ and the index set of basic variables B within PS. A pivot operation tries to improve the current solution using a nonbasic entering variable, say x_l, $l \in N$. The aftermath of this operation is simplified to a properly selected exiting variable and the associated step size ρ is determined by the ratio test. The ratio test is useful on two counts. It maximally exploits the exchange potential of the entering variable and it maintains a basic solution for $\boldsymbol{x}^{k+1}$. The mechanic is incredibly simple although it might sometimes render the linear algebra aspect of the pivot nebulous, especially in the context of degeneracy. In this respect, when PS performs a nondegenerate pivot at iteration $k \geqslant 0$, it goes from vertex $\boldsymbol{x}^k$ represented by a nonoptimal basis to vertex $\boldsymbol{x}^{k+1}$ by moving along an edge (Dantzig and Thapa 2003, Theorem 1.7), a direct consequence of the entering/exiting variable mechanism. In the case of a degenerate pivot, the basis is modified, but the geometrical vertex solution remains the same. In other words, the n-dimensional direction $\vec{\boldsymbol{v}}$ (see Definition 1)

$$\vec{v}_j = \begin{cases} y_j - y_{j+n}, & \forall\, j \in B \cup \{l\} \\ 0, & \forall\, j \in N \backslash \{l\} \end{cases} \tag{9}$$

induced by the selected negative reduced cost entering variable x_l leads outside the domain of *LP* (1) and we do not move. One may want to consider the column $\boldsymbol{a}_l$ of the entering variable as part of the linear span of $\boldsymbol{A}_B$, defined as $\boldsymbol{V}(\boldsymbol{A}_B) := \{\boldsymbol{A}_B \boldsymbol{\lambda} \mid \boldsymbol{\lambda} \in \mathbb{R}^m\}$. By definition, any m-dimensional column belongs to $\boldsymbol{V}(\boldsymbol{A}_B) = \mathbb{R}^m$ meaning in particular that for any nonbasic entering variable

$$\exists!\, \boldsymbol{\lambda} \in \mathbb{R}^m \text{ such that } \sum_{j\in B} \boldsymbol{a}_j \lambda_j = \boldsymbol{a}_l$$
$$\text{which works out to } \boldsymbol{\lambda} = \boldsymbol{A}_B^{-1} \boldsymbol{a}_l. \tag{10}$$

Observe from (9) that a direction in the primal simplex algorithm is not given by the sole entering variable, nor is it limited to the entering/exiting variable couple. It is rather associated with a cycle that combines the entering variable to the affected basic ones. Since the linear combination scalars $\boldsymbol{\lambda}$ in (10) can take any sign, every column of $\boldsymbol{A}_B$ is implicitly expected to have freedom to move in either direction. This *could* unfortunately turn out to be false when the pivot exercise arrives. This possibility can only arise when a nonbasic column is defined by a linear combination containing at least one basic variable at one of its bounds. Indeed, the cycle associated to such an entering variable *might* include a y_j-variable, $j \in B$, with a residual bound of 0, that is, a forward variable $y_j > 0$ with $x_j^k = u_j$ or a backward variable $y_{j+n} > 0$ with $x_j^k = l_j$. The reader may want to compare this with (8) to realize that the degeneracy phenomenon takes an equivalent form in the oracle as well.

2.5. Linear Algebra

Applying the inverse of an arbitrary nonsingular matrix on the equality constraints of *LP* (1) yields an equivalent system. The goal of the linear transformation $\boldsymbol{T}_P^{-1}$ we propose is to structure the equality constraints, where $P \subseteq B$ is an ordered subset of the indices of the basic variables. The set P induces a *subspace basis* $\boldsymbol{A}_P$ with dimension $p := |P|$. In that case, a subset of p rows within $\boldsymbol{A}_P$ are independent. There then exists a row partition $\begin{bmatrix} \boldsymbol{A}_{RP} \\ \boldsymbol{A}_{SP} \end{bmatrix}$ of $\boldsymbol{A}_P$ such that $\boldsymbol{A}_{RP}$ is a nonsingular matrix of size $p \times p$. For instance, an optimal basic solution to the *restricted* phase I problem

$$\min\{\mathbf{1}^\intercal \boldsymbol{\theta} \mid \boldsymbol{A}_P \boldsymbol{x}_P + \boldsymbol{I}_m \boldsymbol{\theta} = \boldsymbol{b}, \boldsymbol{\theta} \geqslant \mathbf{0}\} \tag{11}$$

identifies a set S of rows by associating $(m-p)$ θ-variables with the used columns of $\boldsymbol{I}_m$ yielding the simplex basis $\begin{bmatrix} \boldsymbol{A}_{RP} & \mathbf{0} \\ \boldsymbol{A}_{SP} & \boldsymbol{I}_{m-p} \end{bmatrix}$, hence the requested row partition of $\boldsymbol{A}_P$.

Let $\boldsymbol{V}(\boldsymbol{A}_P) := \{\boldsymbol{A}_P \boldsymbol{\lambda} \mid \boldsymbol{\lambda} \in \mathbb{R}^p\}$ be the vector subspace of $\mathbb{R}^m$ spanned by $\boldsymbol{A}_P$. Because every subset of p linearly independent vectors of $\boldsymbol{V}(\boldsymbol{A}_P)$ can be used as a subspace basis for $\boldsymbol{V}(\boldsymbol{A}_P)$, an alternative set to $\begin{bmatrix} \boldsymbol{A}_{RP} \\ \boldsymbol{A}_{SP} \end{bmatrix}$ is $\begin{bmatrix} \boldsymbol{I}_p \\ \boldsymbol{M} \end{bmatrix}$, where $\boldsymbol{M} := \boldsymbol{A}_{SP} \boldsymbol{A}_{RP}^{-1}$. Together with the $m-p$ independent vectors of $\begin{bmatrix} \mathbf{0} \\ \boldsymbol{I}_{m-p} \end{bmatrix}$, it provides the basis $\boldsymbol{T}_P$ of $\mathbb{R}^m$ and its inverse of particularly simple structure:

$$\boldsymbol{T}_P = \begin{bmatrix} \boldsymbol{I}_p & \mathbf{0} \\ \boldsymbol{M} & \boldsymbol{I}_{m-p} \end{bmatrix} \quad \text{and} \quad \boldsymbol{T}_P^{-1} = \begin{bmatrix} \boldsymbol{I}_p & \mathbf{0} \\ -\boldsymbol{M} & \boldsymbol{I}_{m-p} \end{bmatrix}.$$

Applying $\boldsymbol{T}_P^{-1}$ on the system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ results in $\bar{\boldsymbol{A}} := \boldsymbol{T}_P^{-1}\boldsymbol{A}$ and $\bar{\boldsymbol{b}} := \boldsymbol{T}_P^{-1}\boldsymbol{b}$ as follows:

$$\bar{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A}_R \\ \boldsymbol{A}_S - \boldsymbol{M}\boldsymbol{A}_R \end{bmatrix} \quad \text{and} \quad \bar{\boldsymbol{b}} = \begin{bmatrix} \boldsymbol{b}_R \\ \boldsymbol{b}_S - \boldsymbol{M}\boldsymbol{b}_R \end{bmatrix}. \tag{12}$$

Definition 4 (Gauthier et al. (2016, Proposition 3)). A vector $\boldsymbol{a} \in \mathbb{R}^m$ (and the associated variable, if any) is *compatible with* $\boldsymbol{A}_P$ if and only if $\bar{\boldsymbol{a}}_S := \boldsymbol{a}_S - \boldsymbol{M}\boldsymbol{a}_R = \mathbf{0}$ or, equivalently, $\boldsymbol{a} \in \boldsymbol{V}(\boldsymbol{A}_P)$. Otherwise, the vector $\boldsymbol{a}$ is *incompatible*.

Verifying the equivalence is straightforward when decomposing $\begin{bmatrix} \boldsymbol{a}_R \\ \boldsymbol{a}_S \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{RP} \\ \boldsymbol{A}_{SP} \end{bmatrix} \boldsymbol{\lambda}$. Checking a column vector for compatibility can therefore be done using methods available from the linear algebra arsenal. Some are more efficient than others depending on the content of the matrix $\boldsymbol{A}$, probing cases being the network and set partitioning problems that easily permit the verification of the definition; see the *Transformation matrix insight* paragraph at the end of this section. Compatibility can also be determined over the basis $\boldsymbol{A}_B$ in $O(m^2)$ using the *positive edge* rule (see Towhidi et al. 2014), which reduces the matrix multiplication computational penalty with a stochastic argument.

2.5.1. Column Partition. Given a *subspace basis* $\boldsymbol{A}_P$, let $Q := \{1, \ldots, n\} \backslash P$ be an ordered subset of the indices of all the variables outside the subset $P \subseteq B$. This additional column partition is represented by the matrix $\boldsymbol{A} = [\boldsymbol{A}_P \; \boldsymbol{A}_Q]$. Altogether, we have the row/column partition $\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{RP} & \boldsymbol{A}_{RQ} \\ \boldsymbol{A}_{SP} & \boldsymbol{A}_{SQ} \end{bmatrix}$, where the nonsingular matrix $\boldsymbol{A}_{RP}$ is called the *working basis*. Applying $\boldsymbol{T}_P^{-1}$ on $\boldsymbol{A}$ yields

$$\bar{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{I}_p & \mathbf{0} \\ -\boldsymbol{M} & \boldsymbol{I}_{m-p} \end{bmatrix} \begin{bmatrix} \boldsymbol{A}_{RP} & \boldsymbol{A}_{RQ} \\ \boldsymbol{A}_{SP} & \boldsymbol{A}_{SQ} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{RP} & \boldsymbol{A}_{RQ} \\ \mathbf{0} & \bar{\boldsymbol{A}}_{SQ} \end{bmatrix}. \tag{13}$$

If one thinks of $\boldsymbol{T}_P$ as the current primal simplex basis matrix, the ratio test of an entering variable x_l with null entries in $\bar{\boldsymbol{a}}_{Sl}$ would be performed only on the positive coefficients of $\bar{\boldsymbol{a}}_{Rl}$ and thus only influence variables related to $\boldsymbol{A}_P$. This means that all variables associated with $\boldsymbol{A}_P$ and the row set R are *assumed* to be free whereas all variables associated with $\begin{bmatrix} \mathbf{0} \\ \boldsymbol{I}_{m-p} \end{bmatrix}$ and the row set S are *assumed* to be at their bounds. If the set P indeed corresponds to free variables only ($\boldsymbol{A}_P = \boldsymbol{A}_F$), the resulting step size would be positive for sure. In this spirit, the purpose of $\boldsymbol{T}_P^{-1}$ is to induce a partition in $\bar{\boldsymbol{A}}$ to help look for so-called compatible column vectors; see Section 3.

2.5.2. Transformation Matrix Insight. Ultimately, the transformation matrix produces row and column partitions intimately linked together. Depending on the application, the row partition can even be obtained in the midst of selecting the set P by trying to capture the linear dependence of the technological constraints. Network flow and set partitioning problems are such applications; see Figures 3 and 4, respectively.

In network flows, the free arcs forming $\boldsymbol{A}_F$ can be separated in trees forming a forest. The latter is expressed in matrix form in Figure 3(a) and one can then associate a root node to each tree (in bold). Each of these root nodes corresponds to a linear dependent row in $\boldsymbol{A}_F$ thus forming the row partition presented in

Figure 3. Network Flow Problem Row Partition

(a) Matrix form of the forest induced by $\mathbf{A}_F$

	x_{54}	x_{56}	x_{16}	x_{78}	x_{98}	$x_{8,10}$	x_{72}
5	1	1					
4	−1						
6		−1	−1				
1			**1**				
7				1			1
8				−1	−1	1	
9					1		
10						−1	
2							**−1**
3							

(b) Row partition $\{R, S\}$ of $\mathbf{A}_F$

		x_{54}	x_{56}	x_{16}	x_{78}	x_{98}	$x_{8,10}$	x_{72}
	5	1	1					
	4	−1						
	6		−1	−1				
	7				1			1
R	8				−1	−1	1	
	9					1		
	10						−1	
	1			**1**				
S	2							**−1**
	3							

Figure 3(b). A constructive approach leading to a contracted network for the identification of negative cycles is presented in Gauthier et al. (2017).

In set partitioning problems, and more specifically when using DCA, a subspace basis $\left[\begin{smallmatrix} I_p \\ M \end{smallmatrix}\right]$ is obtained on the fly while *heuristically* trying to select linear independent rows within $\boldsymbol{A}_F$. This process is sketched in Figure 4. In Figure 4(a), the original matrix $\boldsymbol{A}_F$ is presented whereas Figure 4(b) reorganizes the duplicated rows on the bottom (and this reorganization then applies to the original system). By associating a unique identifier to each singled out row in the top portion and replicating these identifiers in the bottom portion, Figure 4(b) obtains five rows in the set R and three in the set S. Figure 4(c) uses these identifiers by replacing the matrix content with trivial unit references for each identifier, thus obtaining the subspace basis $\left[\begin{smallmatrix} I_5 \\ M \end{smallmatrix}\right]$. One can easily verify that the four columns of $\boldsymbol{A}_F$ in Figure 4(b) belong to the span of that subspace basis. This is also true for a fifth column from the simplex basis $\boldsymbol{A}_B$, its actual content being irrelevant.

3. Vector Space Decomposition Framework

In this section, we look at the essential components of the proposed framework. The algorithm relies on an oracle to iterate. The latter is dynamically updated with respect to the values of the current solution $\boldsymbol{x}^k$. As such, the oracle is produced by applying a row/column partition, based on the transformation matrix, on the aforementioned residual problem. In a nutshell, the portions obtained from this partition communicate with each other in the same way a master/subproblem would. In practice, we capitalize on the dynamic partition by treating its content like a Dantzig–Wolfe decomposition; see Dantzig and Wolfe (1960). The generic framework is presented in Figure 5 as a concluding overview, whereas the different steps listed therein are regrouped in four parts. Section 3.2 details the transformation leading to the pricing oracle expressed in Section 3.3. An optimal solution of the latter has to be manipulated to obtain a so-called *lifted cycle* as presented in Section 3.4. We finalize the iteration with update operations discussed in Section 3.5.

3.1. Initialization

It all starts at iteration $k = 0$ with some basic feasible solution $\boldsymbol{x}^0$ and column partition $\{F, L, U\}$. The construction of the residual problem $LP(\boldsymbol{x}^k)$ (2) calls for a change of variables: $y_j - y_{j+n} := x_j - x_j^k$, $y_j\, y_{j+n} = 0$, $\forall\, j \in \{1, \ldots, n\}$. These variables are bounded by $0 \leqslant y_j \leqslant r_j^k$, $\forall\, j \in \{1, \ldots, 2n\}$.

Figure 4. Set Partitioning Problem Row Partition

(a) Original matrix $\boldsymbol{A}_F$

$$\begin{pmatrix} 1 & 1 & & \\ & 1 & 1 & \\ 1 & 1 & & \\ & & 1 & 1 \\ 1 & & 1 & \\ 1 & & & 1 \\ & 1 & 1 & \\ 1 & & & 1 \end{pmatrix}$$

(b) Row partition $\left[\begin{smallmatrix} A_{RF} \\ A_{SF} \end{smallmatrix}\right]$

$$\left(\begin{array}{cccc} 1 & 1 & & \\ & 1 & 1 & \\ & & 1 & 1 \\ 1 & & 1 & \\ 1 & & & 1 \\ \hline 1 & 1 & & \\ & 1 & 1 & \\ 1 & & & 1 \end{array}\right) \begin{matrix} a \\ b \\ c \\ d \\ e \\ \hline a \\ b \\ e \end{matrix}$$

(c) Subspace basis $\left[\begin{smallmatrix} I_5 \\ M \end{smallmatrix}\right]$

$$\left(\begin{array}{ccccc} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \\ \hline 1 & & & & \\ & 1 & & & \\ & & & & 1 \end{array}\right)$$

Figure 5. Generic Vector Space Decomposition Framework for Linear Programs

Initialization: Iteration $k := 0$;
Feasible solution $\boldsymbol{x}^0$;
Select the subspace basis $\boldsymbol{A}_P$;
1: Derive the matrix $\boldsymbol{T}_P^{-1}$ and the row/column partitions $\{R,S\}/\{P,Q\}$ of $\boldsymbol{A}$;
2: Divide the vector $\boldsymbol{y} \in \mathbb{R}_+^{2n}$ with the three sets H_P, V_P and Z_P as in (14)–(16);
3: Solve the pricing problem (21) for μ_V^k and the projected cycle W_V^k;
4: If $\mu_V^k \geqslant 0$, **terminate** with an optimality certificate for $\boldsymbol{x}^k$;
5: Extract the lifted cycle $W_{V\cup H}^k$ from (22)–(23) and the direction $\vec{\boldsymbol{v}}^k$;
6: Compute the step size ρ^k (24);
7: Obtain the solution $\boldsymbol{x}^{k+1} := \boldsymbol{x}^k + \rho^k \vec{\boldsymbol{v}}^k$ using the system (25);
8: Update the subspace basis $\boldsymbol{A}_P$;
9: Iterate $k := k+1$;

3.2. Structured Residual Problem

Once an arbitrary subspace basis $\boldsymbol{A}_P$ is selected, the induced matrix $\boldsymbol{T}_P^{-1}$ is derived along with a row/column partition $\{R,S\}/\{P,Q\}$ of matrix $\boldsymbol{A}$. The only point we shall insist on is the structuring effect of $\boldsymbol{T}_P^{-1}$. The same can obviously be observed in the residual problem $LP(\boldsymbol{x}^k)$. Let us divide the vector $\boldsymbol{y} \in \mathbb{R}_+^{2n}$ according to P with the three sets

$$H_P \equiv H_P(\boldsymbol{x}^k) := \bigcup_{j\in P} \{j, j+n\} \tag{14}$$

$$V_P \equiv V_P(\boldsymbol{x}^k) := J^k \backslash H_P \tag{15}$$

$$Z_P \equiv Z_P(\boldsymbol{x}^k) := \{1,\dots,2n\}\backslash(H_P \cup V_P), \tag{16}$$

where the y-variables in the set H_P are *hidden* from the oracle, the residual variables in V_P are rather *visible* in the oracle, whereas the remaining ones (with null residual upper bounds) in Z_P remain at zero. The residual y-variables are obviously exhaustively considered within H_P and V_P. However, observe that while V_P contains only residual variables, H_P may or may not. That is, if there exists a $j \in P$ such that either $j \in L$ or $j \in U$, then y_{j+n} or, respectively, y_j has a null residual capacity. Discarding the variables in Z_P from the residual problem, that is, $y_j^k = 0, \forall\, j \in Z_P$, the formulation (2) can then be rewritten as

$$\begin{aligned} z^\star = z^k + \min\ & \sum_{j\in H_P} d_j y_j + \sum_{j\in V_P} d_j y_j \\ \text{s.t.}\ & \sum_{j\in H_P} \boldsymbol{k}_{Rj} y_j + \sum_{j\in V_P} \boldsymbol{k}_{Rj} y_j = \boldsymbol{0} \quad [\boldsymbol{\psi}_R] \\ & \sum_{j\in V_P} \bar{\boldsymbol{k}}_{Sj} y_j = \boldsymbol{0} \quad [\boldsymbol{\psi}_S] \\ & 0 \leqslant y_j \leqslant r_j^k, \quad \forall\, j \in H_P \cup V_P, \end{aligned} \tag{17}$$

where $\boldsymbol{k}_j$ is the jth column vector of $\boldsymbol{K}$, $\bar{\boldsymbol{K}} := \boldsymbol{T}_P^{-1}\boldsymbol{K}$, and $\boldsymbol{\psi}^\intercal = [\boldsymbol{\psi}_R^\intercal,\ \boldsymbol{\psi}_S^\intercal]$ is the vector of dual variables of the transformed system. The original dual vector $\boldsymbol{\pi}$ can be retrieved from $\boldsymbol{\psi}$ using the expression $\boldsymbol{\pi}^\intercal = \boldsymbol{\psi}^\intercal \boldsymbol{T}_P^{-1}$, that is, $[\boldsymbol{\pi}_R^\intercal, \boldsymbol{\pi}_S^\intercal] = [\boldsymbol{\psi}_R^\intercal - \boldsymbol{\psi}_S^\intercal \boldsymbol{M}, \boldsymbol{\psi}_S^\intercal]$.

3.3. Pricing Oracle: Cycle and Direction

The pricing problem exploits the structure in (17) and derives an oracle based on the resulting transformation. The oracle is presented in both primal and dual forms, each having its own interpretation. Let us start with the dual form that is derived by trying to meet the necessary and sufficient optimality conditions. That is, if the assumed free variables in the set P are at an optimal value, the dual variables of $\boldsymbol{\pi}$, or those of $\boldsymbol{\psi}$, must impose a reduced cost of zero on these variables [complementary slackness conditions (4)]:

$$\begin{aligned} \boldsymbol{0} = \bar{\boldsymbol{c}}_P^\intercal &= \boldsymbol{c}_P^\intercal - \boldsymbol{\psi}_R^\intercal \boldsymbol{A}_{RP} \\ (&= \boldsymbol{c}_P^\intercal - (\boldsymbol{\pi}_R^\intercal + \boldsymbol{\pi}_S^\intercal \boldsymbol{M})\boldsymbol{A}_{RP} = \boldsymbol{c}_P^\intercal - \boldsymbol{\pi}_R^\intercal \boldsymbol{A}_{RP} - \boldsymbol{\pi}_S^\intercal \boldsymbol{A}_{SP}). \end{aligned} \tag{18}$$

Furthermore, if the current solution $\boldsymbol{x}^k$ is optimal, there exists a dual vector $\boldsymbol{\psi}_S$ such that the smallest reduced cost of the remaining variables in V_P, say μ_V, is nonnegative [dual condition (6)]. This verification can be done with the linear program

$$\begin{aligned} \max\ & \mu_V \\ \text{s.t.}\ & \mu_V \leqslant \bar{d}_j = d_j - \boldsymbol{\psi}_R^\intercal \boldsymbol{k}_{Rj} - \boldsymbol{\psi}_S^\intercal \bar{\boldsymbol{k}}_{Sj} \quad [y_j]\ \forall\, j \in V_P, \end{aligned} \tag{19}$$

where the vector $\boldsymbol{\psi}_R^\intercal = \boldsymbol{c}_P^\intercal \boldsymbol{A}_{RP}^{-1}$ is fixed by (18), whereas the vector $\boldsymbol{\psi}_S^\intercal$ is part of the optimization so as to maximize the minimum reduced cost. More generally, one can use the scalars $w_j > 0, \forall\, j \in V_P$, and maximize the minimum normalized reduced cost (see Section 3.3.1):

$$\begin{aligned} \max\ & \mu_V \\ \text{s.t.}\ & \mu_V \leqslant \frac{\bar{d}_j}{w_j} = \frac{1}{w_j}(d_j - \boldsymbol{\psi}_R^\intercal \boldsymbol{k}_{Rj} - \boldsymbol{\psi}_S^\intercal \bar{\boldsymbol{k}}_{Sj}) \\ & \qquad\qquad [y_j] \quad \forall\, j \in V_P. \end{aligned} \tag{20}$$

Dualizing (20), we obtain the primal form that comprises $m - p + 1$ constraints:

$$\begin{aligned} \min\ & \sum_{j\in V_P} \tilde{d}_j y_j \\ \text{s.t.}\ & \sum_{j\in V_P} \bar{\boldsymbol{k}}_{Sj} y_j = \boldsymbol{0} \quad [\boldsymbol{\psi}_S] \\ & \sum_{j\in V_P} w_j y_j = 1 \quad [\mu_V] \\ & y_j \geqslant 0, \quad \forall\, j \in V_P, \end{aligned} \tag{21}$$

where $\tilde{d}_j := d_j - \boldsymbol{\psi}_R^\intercal \boldsymbol{k}_{Rj} = d_j - \boldsymbol{c}_P^\intercal \boldsymbol{A}_{RP}^{-1} \boldsymbol{k}_{Rj}, \forall\, j \in V_P$. The oracle interpretation is done through the primal-dual pair (20)–(21). It brings together the negative cycles and $\boldsymbol{T}_P^{-1}$.

3.3.1. Oracle Interpretation.

First of all, the formulation (21) is always feasible unless $\boldsymbol{x}^0$ is the only feasible solution of LP (1). Reciprocally, (20) is always feasible although unbounded in the exception case. Note that we can ensure that the primal-dual pricing system is feasible/bounded if we write the normalization

constraint as a less-than-or-equal inequality or equivalently one imposes $\mu_V \leqslant 0$. Furthermore, weight values of $\boldsymbol{w}$ used for the normalization constraint can be set in stone or updated dynamically. In the former case, think of the all-ones vector typically used in network flows (see Gauthier et al. 2015) or the norm based weights such as $w_j = w_{j+n} = \|\boldsymbol{a}_j\|$, $\forall\, j \in \{1,\ldots,n\}$, which makes the ratio $\bar{d}_j/w_j$ impervious to the scaling of variable x_j. In the latter case, dynamic weight choices can also be made to help steer the pricing problem toward or away from certain solutions; see Rosat et al. (2017a, b) for several alternatives that have been particularly successful for solving set partitioning problems using the *integral simplex using decomposition* algorithm (ISUD) (Zaghrouti et al. 2014). Finally, it can also be noted that all other things being equal, a smaller value of w_j favors the selection of variable y_j in the pricing problem. Conversely, an infinite weight is equivalent to discarding a variable in a partial pricing strategy.

Let $\boldsymbol{y}_V^k := [y_j^k]_{j\in V_P}$ denote an optimal solution to the primal-dual system of value μ_V^k at iteration $k \geqslant 0$. If $\mu_V^k \geqslant 0$, the dual optimality conditions are satisfied and we terminate with an optimal solution $\boldsymbol{x}^k$. Otherwise, $\mu_V^k < 0$ and the current solution might be *improved* by following a direction.

3.4. Projected and Lifted Cycles

An optimal negative cycle, say W_V^k, derived from (21) is augmenting since only residual variables y_j, $j \in V_P$, are considered (see Definition 3). However, the fact that the solution $\boldsymbol{y}_V^k$ for the visible variables is built omitting the variables in $\boldsymbol{y}_H \geqslant \boldsymbol{0}$ means that it only provides a *portion* of $\boldsymbol{y}^k$. Then again, by construction of the linear transformation, these hidden components are uniquely determined within (17) in the system of row set R where the free nature of variables in P is assumed, that is,

$$\sum_{j\in H_P} \boldsymbol{k}_{Rj} y_j + \sum_{j\in V_P} \boldsymbol{k}_{Rj} y_j^k = \boldsymbol{0}, \quad y_j \geqslant 0,\ \forall\, j \in H_P. \tag{22}$$

The nonsingularity of A_{RP} is made apparent in the transformed system (13) and also finds its way in the equivalent formulation (17). The solution $\boldsymbol{y}_H^k$ to (22) is then determined alike (10) by $\boldsymbol{\lambda}_P := \boldsymbol{A}_{RP}^{-1}(-\sum_{j\in V_P} \boldsymbol{k}_{Rj} y_j^k)$ as

$$y_j^k = \begin{cases} -\lambda_j, & \text{if } \lambda_j < 0 \\ 0, & \text{otherwise} \end{cases} \quad \text{and}$$

$$y_{j+n}^k = \begin{cases} \lambda_j, & \text{if } \lambda_j > 0 \\ 0, & \text{otherwise} \end{cases} \quad \text{for all } j \in P, \tag{23}$$

and the direction $\vec{\boldsymbol{v}}^k$ follows from the application of Definition 1 to $\boldsymbol{y}^k = [\boldsymbol{y}_H^k, \boldsymbol{y}_V^k, \boldsymbol{y}_Z^k]$. Observe that the complementarity condition $y_j\, y_{j+n} = 0$, $\forall\, j \in \{1,\ldots,n\}$, is taken into account at every stage. First off, by looking for extreme point solutions to the pricing problem, any negative cycle W_V^k cannot contain both y_j and y_{j+n} variables simultaneously. Secondly, $\boldsymbol{y}_H^k$ is established in (23) by dichotomy on the signs of $\boldsymbol{\lambda}_P$.

All in all, the cycle W_V^k found in the pricing problem is the support of incomplete information about the direction yet, once W_V^k is identified, the complete cycle is always uniquely determined. In this respect, let W_V^k be called a *projected cycle* on the visible variables in the linear system (21) whereas the full cycle produced with $[\boldsymbol{y}_H^k, \boldsymbol{y}_V^k]$ is named the *lifted cycle* and is denoted by $W_{V\cup H}^k$.

Furthermore, since the reduced cost of the hidden variables is $\bar{d}_j = 0$, $\forall\, j \in H_P$, we must have $\bar{d}(W_V^k) = \bar{d}(W_{V\cup H}^k)$. By Lemma 1, the reduced cost of the cycle $W_{V\cup H}^k$ also corresponds to its cost such that fixing the reduced costs of the hidden components to zero transfers the reduced cost information to the visible variables as $\tilde{d}_j$, $j \in V_P$, although the latter contains some dual variables free to be optimized in the pricing problem. Finally, recall that the projected cycle W_V^k is always augmenting. Whether or not the lifted cycle $W_{V\cup H}^k$ is itself guaranteed to be augmenting over the set of residual variables in J^k is directly related to the free nature of the hidden variables in H_P. In other words, whether the step size computed next is certainly positive depends on the content of P (see Proposition 2).

3.5. Step Size and Updates

The step size ρ^k of the lifted cycle $W_{V\cup H}^k$ is computed with respect to the residual capacities of the variables forming it, divided by their respective contribution as

$$\rho^k := \min_{j\in W_{V\cup H}^k} \left\{ \frac{r_j^k}{y_j^k} \right\} \geqslant 0. \tag{24}$$

A new primal solution $\boldsymbol{x}^{k+1} := \boldsymbol{x}^k + \rho^k\, \vec{\boldsymbol{v}}^k$ with cost $z^{k+1} := z^k + \rho^k\, \mu_V^k$ is obtained as

$$\forall\, j \in \{1,\ldots,n\}, \quad x_j^{k+1} := \begin{cases} x_j^k + \rho^k y_j^k, & \text{if } j \in W_{V\cup H}^k \\ x_j^k - \rho^k y_j^k, & \text{if } j+n \in W_{V\cup H}^k \\ x_j^k, & \text{otherwise.} \end{cases} \tag{25}$$

Depending on the choice of the subspace basis $\boldsymbol{A}_P$, $\boldsymbol{x}^{k+1}$ represented by $[\boldsymbol{x}_F^{k+1}, \boldsymbol{x}_L^{k+1}, \boldsymbol{x}_U^{k+1}]$ could be nonbasic. Section 4.2 explains how and when this can happen with the conceptualization of *interior directions*. We simply mention that any nonbasic solution $\boldsymbol{x}^{k+1}$ can be rendered basic by solving a restricted problem over the set of free variables:

$$\begin{aligned} z^{k+1} = \min\ & \boldsymbol{c}_F^\intercal \boldsymbol{x}_F + \boldsymbol{c}_L^\intercal \boldsymbol{x}_L + \boldsymbol{c}_U^\intercal \boldsymbol{x}_U \\ \text{s.t.}\ & \boldsymbol{A}_F \boldsymbol{x}_F + \boldsymbol{A}_L \boldsymbol{x}_L + \boldsymbol{A}_U \boldsymbol{x}_U = \boldsymbol{b} \\ & \boldsymbol{l}_F \leqslant \boldsymbol{x}_F \leqslant \boldsymbol{u}_F, \quad \boldsymbol{x}_L = \boldsymbol{x}_L^{k+1}, \quad \boldsymbol{x}_U = \boldsymbol{x}_U^{k+1}. \end{aligned} \tag{26}$$

This problem identifies augmenting cycles comprising free variables only, and increases or decreases the value of these variables until some lower and upper bounds are reached while possibly improving the overall solution cost. In network flows terminology, one obtains a *cycle free solution*, that is, a network solution containing no cycle composed of free arcs only; see Ahuja et al. (1993).

There only remains to update the residual problem $LP(\boldsymbol{x}^{k+1})$ with residual capacities $\boldsymbol{r}^{k+1}$ and column partition $\{F, L, U\}$, and to select a new subspace basis $\boldsymbol{A}_P$. Another iteration $k \longrightarrow k+1$ then starts in Step 1.

4. Properties

The generic framework ultimately depends on a single parameter, that is, the selection of the set P. Section 4 derives two propositions revolving around this selection. In Section 4.1, we underline particular well-known variants of this framework, whereas Section 4.2 qualifies the kinds of directions found in accordance with the selected set P. In Section 4.3, an illustrative example on a three-dimensional polyhedron shows that a direction with a positive step size can occur on an edge, or be interior. Finally, Section 4.4 draws the line between the proposed framework and actual implementations based on its principles.

4.1. Special Cases

Let us start with a family of variants that perform a positive step size at every iteration. This is completed with four specific variants found in the linear programming literature.

Proposition 2. *Let $\boldsymbol{x}^k$, $k \geqslant 0$, be a nonoptimal basic solution to LP (1). Given $P \subseteq F$, the step size of $\vec{\boldsymbol{v}}^k$ is guaranteed to be positive.*

Proof. If $P \subseteq F$, or equivalently $P \cap (L \cup U) = \varnothing$, then all the hidden y-variables in H_P are residual as well as those in V_P. Therefore, the primal-dual pair of the pricing problem (20)–(21), respectively, matches the necessary and sufficient primal-dual optimality conditions of Proposition 1. Indeed, the a posteriori lifted cycle $W^k_{V\cup H}$ obtained from W^k_V trivially only uses variables in J^k and is as such a negative cycle in $LP(\boldsymbol{x}^k)$, meaning that the associated step size is positive. □

Remark. Consider the case $P \nsubseteq F$, or equivalently $P \cap (L \cup U) \neq \varnothing$, from the perspective of dual optimality conditions. If there exists a $j \in P$ such that $j \notin F$ (i.e., $j \in L \cup U$), then the reduced cost of zero imposed on *both*, y_j and y_{j+n} is too stringent. The reduced cost of a y-variable with a null residual capacity is irrelevant, which would incidentally have granted more freedom to $\boldsymbol{\psi}_R$. One can see the pricing oracle construction also misleads the verification of the dual condition (6). Indeed, (18) implies $\bar{d}_j = \bar{d}_{j+n} = 0$, $\forall\, j \in P$, or equivalently $\bar{d}_j = d_j - \boldsymbol{\psi}_R^\intercal \boldsymbol{k}_{Rj} \geqslant 0$, $\forall\, j \in H_P$, that is, the nonfree variables contained in H_P are all part of the verification when they should not. In other words, this overly restrictive observation is also echoed in the primal form where a cycle using such a y-variable is made possible, that is, the additional column in the primal form comes from the additional constraint in the dual form. Observe that this is portrayed in PS when facing a degenerate basis.

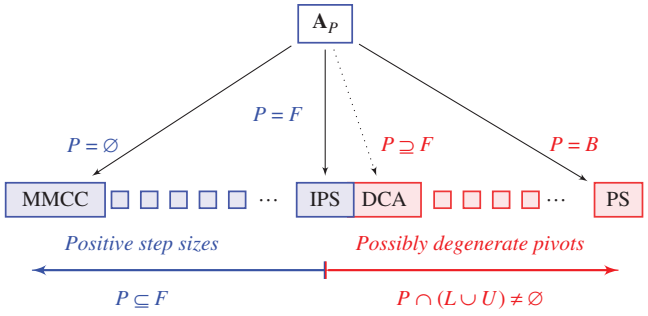
Case $P = \varnothing$. When choosing $P = \varnothing$, it amounts to a subspace basis $\boldsymbol{A}_\varnothing$ of dimension zero, which in turn means that $V(\boldsymbol{A}_\varnothing) = \{\boldsymbol{0}\}$. Since the vector subspace contains only the zero vector, there are no compatible variables, basic or otherwise. The transformation is trivial, that is, $\boldsymbol{T}_\varnothing = \boldsymbol{T}_\varnothing^{-1} = \left[\begin{smallmatrix} I_0 & \varnothing \\ \varnothing & I_m \end{smallmatrix}\right] = \boldsymbol{I}_m$. From a dual point of view, the entire m-dimensional dual vector $\boldsymbol{\pi}$ is optimized to maximize the minimum reduced cost. From a primal point of view, the pricing problem contains precisely all the residual variables and guarantees a positive step size (otherwise the current solution is optimal). When $\boldsymbol{A}$ is the node-arc incidence matrix of a network, this particular case corresponds to the remarkable strongly polynomial minimum mean cycle-canceling algorithm (MMCC) of Goldberg and Tarjan (1989) devised for capacitated minimum cost flow problems. With respect to arbitrary linear programs, it appears natural to think of yet another analogy: *minimum weighted cycle-canceling* algorithm. From a mechanical point of view, the adaptation is straightforward. However, the extent to which time complexity properties are transferable is left for another paper.

Case $P = F$. Choosing $P = F$ corresponds to the strategy developed by Elhallaoui et al. (2011) in IPS. The vector subspace $\boldsymbol{V}(\boldsymbol{A}_F)$ includes all columns of $\boldsymbol{A}_F = \left[\begin{smallmatrix} \boldsymbol{A}_{RF} \\ \boldsymbol{A}_{SF} \end{smallmatrix}\right]$ but none associated with the basic variables at one of their bounds: $\forall\, j \in B$, $\boldsymbol{a}_j \in \boldsymbol{V}(\boldsymbol{A}_F) \Leftrightarrow j \in F$. The transformation is given by $\boldsymbol{T}_F^{-1} = \left[\begin{smallmatrix} \boldsymbol{I}_f & \boldsymbol{0} \\ -\boldsymbol{M} & \boldsymbol{I}_{m-f} \end{smallmatrix}\right]$, where $\boldsymbol{M} = \boldsymbol{A}_{SF}\boldsymbol{A}_{RF}^{-1}$. As a special case of Proposition 2, a positive step size is guaranteed.

Case $P = B$. When choosing $P = B$, we have m linearly independent column vectors with $\boldsymbol{A}_B$ and $\boldsymbol{V}(\boldsymbol{A}_B) = \mathbb{R}^m$. All variables are compatible whereas the sets P and Q, respectively, correspond to all basic and nonbasic variables. The subspace basis induces $\boldsymbol{T}_B = \boldsymbol{T}_B^{-1} = \left[\begin{smallmatrix} I_m & \varnothing \\ \varnothing & I_0 \end{smallmatrix}\right] = \boldsymbol{I}_m$ yielding once again a trivial transformation. Most importantly, it fixes $\boldsymbol{\pi}^\intercal = \boldsymbol{c}_B^\intercal \boldsymbol{A}_B^{-1}$, that is, *all* dual variables, and with $\boldsymbol{w} = \boldsymbol{1}$, the generic framework becomes the primal simplex algorithm with Dantzig's pivot-selection rule. When $B \cap (L \cup U) \neq \varnothing$, that is, when at least one basic degenerate variable is present, the set H_P contains y-variables with null residual capacities and a null step size can occur, that is, a degenerate pivot.

Case $P \supseteq F$. When choosing $P \supseteq F$, we have $f \leqslant p \leqslant m$ linearly independent column vectors in $\boldsymbol{A}_P$. This case

Figure 6. (Color online) Special Cases of VSD: MMCC (1989), IPS (2008), DCA (2005), and PS (1947)



notably includes DCA, the strategy used in Elhallaoui et al. (2005) for solving set partitioning problems by column generation. While the equivalent form with the columns of $\boldsymbol{A}_B$ exists, the subspace basis $\left[\begin{smallmatrix} I \\ M \end{smallmatrix}\right]$ is obtained by design, more precisely heuristic row clustering as seen in Figure 4. If $P \supset F$, then the set H_P contains $p - f$ y-variables with null residual capacities. This possibly larger than necessary subspace basis gives a lot of freedom in the implementation of DCA, a method steered by practical imperatives.

Figure 6 synthesizes these special cases with respect to Proposition 2, providing also the year they have been designed.

4.2. Interior Directions

Since PS relies on the edge movement induced by a pivot by considering the direction of travel, let us add a layer of definition on the resulting impact of this direction.

Definition 5. Let C be a polyhedron. Given a vertex $\boldsymbol{x} \in C$ and a direction $\vec{\boldsymbol{v}} \neq \boldsymbol{0}$, let $\boldsymbol{x} + \rho\vec{\boldsymbol{v}} \in C$ for some $\rho > 0$. The vector $\vec{\boldsymbol{v}}$ is called an *edge direction* originating from $\boldsymbol{x}$ if for $0 < \delta < \rho$, the vector $\boldsymbol{x} + \delta\vec{\boldsymbol{v}}$ belongs to an edge of C. Otherwise, a nonedge direction is called an *interior direction* originating from $\boldsymbol{x}$.

An important property of IPS (where $P = F$) is its movement on an edge (Elhallaoui et al. 2011, Proposition 4). Proposition 2 shows that the family of algorithms with $P \cap (L \cup U) = \varnothing$, or equivalently $P \subseteq F$, ensures a positive step size at every iteration. This also means that the oracle associated with any of these variants is able to verify the necessary and sufficient optimality conditions. While one might rest uneasy about equivalent necessary and sufficient optimality conditions provided by two different oracles, the following proposition sheds light on their content and characterizes improving interior directions originating from a nonoptimal basic solution $\boldsymbol{x}^k$. In a nut shell, since the case $P = F$ only identifies edge directions, it provides the smallest dimensional cone $\bar{K}_{SV}\boldsymbol{y}_V = \boldsymbol{0}$, $\boldsymbol{y} \geqslant \boldsymbol{0}$, able to exhaustively identify the set of feasible edge directions. Variants using $P \subset F$ must then contain these edge directions or combinations of these, that is, interior directions.

Proposition 3. *Let $\boldsymbol{x}^k$, $k \geqslant 0$, be a nonoptimal basic solution to LP (1). For $P \subset F$, an interior direction $\vec{\boldsymbol{v}}^k$ (if any) is a nonnegative combination of the edge directions.*

Proof. For $P = F$, any lifted solution $\boldsymbol{y}^k = [\boldsymbol{y}_V^k, \boldsymbol{y}_H^k, \boldsymbol{y}_Z^k]$ to (21)–(23) is in a one-to-one correspondence with an extreme ray of the cone defined by removing the normalization constraint. Let Ω_F^k be the index set of these extreme rays, indexed by ω. Any solution $\boldsymbol{y}$ to (21)–(23) is nonnull and by the representation theorems of Minkowski and Weyl (see Section 7.2 in Schrijver 1986), it can then be expressed as a nonnegative combination of these extreme rays, that is, $\boldsymbol{y} = \sum_{\omega\in\Omega_F^k} \boldsymbol{y}^\omega \lambda^\omega$, $\lambda^\omega \geqslant 0$, $\forall\, \omega \in \Omega_F^k$, or component-wise for $j \in \{1, \ldots, 2n\}$, $y_j = \sum_{\omega\in\Omega_F^k} y_j^\omega \lambda^\omega$, $\lambda^\omega \geqslant 0$, $\forall\, \omega \in \Omega_F^k$. By Definition 1, every component $\vec{v}_j$, $j \in \{1, \ldots, n\}$, of the associated direction $\vec{\boldsymbol{v}}$ is given by

$$\vec{v}_j = y_j - y_{j+n} = \sum_{\omega\in\Omega_F^k} (y_j^\omega - y_{j+n}^\omega)\lambda^\omega, \quad \lambda^\omega \geqslant 0, \forall\, \omega \in \Omega_F^k,$$

and hence, $\vec{\boldsymbol{v}}$ is a nonnegative combination of the edge directions:

$$\vec{\boldsymbol{v}} = \sum_{\omega\in\Omega_F^k} \boldsymbol{v}^\omega \lambda^\omega, \quad \lambda^\omega \geqslant 0, \forall\, \omega \in \Omega_F^k. \tag{27}$$

For $P \subset F$, the pricing problem involves more visible y-variables compared to the case with $P = F$ because $V_P \supset V_F$. At the same time, it contains $f - p$ more constraints since $m - p + 1 > m - f + 1$. Therefore, if $\vec{\boldsymbol{v}}^k$ is not an edge direction, then it is interior and by (27) it can be expressed as a nonnegative combination of the edge directions. □

Note that a direction leading to a nonbasic solution must be an interior direction. As such, observe that the case $P = \varnothing$ is one of the variants susceptible to lead to nonbasic solutions. However, since $\boldsymbol{T}_\varnothing^{-1} = \boldsymbol{I}_m$ and all dual variables are optimized in the pricing problem, the simplifications applicable to the different steps in Figure 5 imply that maintaining the basic nature of encountered solutions is irrelevant. For all other cases, fetching an index set P of linearly independent columns can be made simply by solving (26).

4.3. Illustrative Examples

Consider the linear program expressed in (28) with x_1, x_2, and x_3, and four inequality constraints. Let $x_4, \ldots, x_7$ be the slack variables associated with each constraint and assume the initial basic solution $\boldsymbol{x}^0$ uses these at values $x_4^0 = 21$, $x_5^0 = 8$, $x_6^0 = 15$ and $x_7^0 = 32$ for a total cost of $z^0 = 0$. This basic solution is nondegenerate

Figure 7. Extreme Rays $\boldsymbol{y}_V$ at $\boldsymbol{x}^0$ in Pricing for $P = F$ and $P = \varnothing$

Variants	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}
$P = F$	1	0	0											
	0	1	0											
	0	0	1											
$P = \varnothing$	1	0	0	0	1	0	1				2	0	2	0
	0	1	0	1	0	1	1				0	1	0	0
	0	0	1	0	1	1	0				2	0	0	2
	0	2	1	0	0	3	0				0	1	0	0
	1	2	0	0	0	0	3				0	1	0	0
	2	3	1	0	0	0	3				3	0	0	0

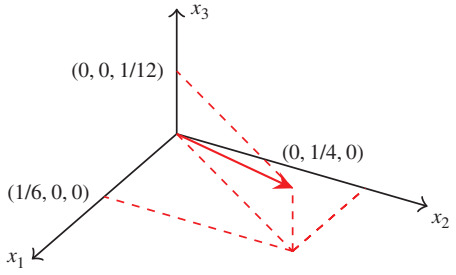
and hence there are three edge directions according to the selected entering variable x_1, x_2, or x_3.

$$\begin{aligned} \max\ & 130x_1 + 80x_2 + 60x_3 \\ \text{s.t.}\ & 2x_1 - x_2 + 2x_3 \leqslant 21 \\ & -x_1 + x_2 - x_3 \leqslant 8 \\ & 2x_1 - x_2 - x_3 \leqslant 15 \\ & -x_1 - x_2 + 2x_3 \leqslant 32 \\ & x_1, x_2, x_3 \geqslant 0. \end{aligned} \tag{28}$$

Let us start with the possible values of the projected vector $\boldsymbol{y}_V \in \mathbb{R}_+^{|V|}$ arising from $\boldsymbol{y} \in \mathbb{R}_+^{14}$ while solving the oracle (21). For $P = F$, eight variables are hidden from the pricing problem, that is, the forward and backward y-variables associated with the four slack variables in F, only three are visible in (21), that is, y_1, y_2 and y_3, whereas y_8, y_9 and y_{10} are fixed to zero. The three-dimensional extreme ray (y_1, y_2, y_3) can take values $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$. For $P = \varnothing$ and incidentally $H_P = \varnothing$, y_8, y_9 and y_{10} are again fixed to zero such that all the residual variables are visible in the oracle. The vector $\boldsymbol{y}_V$ is therefore the 11-dimensional vector $(y_1, \ldots, y_7, y_{11}, \ldots, y_{14})$ for which the six extreme rays, conveniently expressed with integers, are listed in Figure 7.

Figure 8 details the content of $\vec{\boldsymbol{v}}^0 \in \mathbb{R}^7$ found under the suggested set P and weight vector $\boldsymbol{w}$. One would of course use the stopping criterion $\mu_V^k \leqslant 0$ since (28) is being maximized. The minimum cost of the associated cycle W_V^0 is given by μ_V^0, the step size ρ^0 recovered

Figure 9. (Color online) Three-Dimensional Interior Direction $(\vec{v}_1^0, \vec{v}_2^0, \vec{v}_3^0) = (\frac{1}{6}, \frac{1}{4}, \frac{1}{12})$ with $P = \varnothing$ and $\boldsymbol{w} = \mathbf{1}$



with (24), and the new solution $\boldsymbol{x}^1$ obtained with (25). As expected, the direction $\vec{\boldsymbol{v}}^0 = (1,0,0,-2,1,-2,1)$ found with IPS ($P = F$) follows an edge and yields an extreme point solution at $\boldsymbol{x}^1$. Notice that since $\boldsymbol{x}^0$ is nondegenerate, we have $F = B$ and we would have found the same direction upon selecting x_1 as the entering variable in PS.

When $P = \varnothing$ and $w_j = 1$, $\forall j \in \{1, \ldots, 14\}$, $\vec{\boldsymbol{v}}^0 = (\frac{1}{6}, \frac{1}{4}, \frac{1}{12}, -\frac{1}{4}, 0, 0, \frac{1}{4})$ induced by the last extreme ray of Figure 7 so happens to be interior (Figure 9) and yields a feasible solution $\boldsymbol{x}^1$, which is nonbasic (six variables take positive values). By Proposition 3, this $\vec{\boldsymbol{v}}^0$ is indeed the combination of edge directions, that is,

$$\begin{aligned} \vec{\boldsymbol{v}}^0 = {} & \tfrac{1}{6}(1,0,0,-2,1,-2,1) + \tfrac{1}{4}(0,1,0,1,-1,1,1) \\ & + \tfrac{1}{12}(0,0,1,-2,1,1,-2). \end{aligned}$$

The last example in Figure 8 still uses the set $P = \varnothing$ but uses a weight vector whose every element w_j is determined by computing the squared norm $\|\boldsymbol{a}_j\|^2$ of each column, that is, $w_1 = 2^2 + (-1)^2 + 2^2 + (-1)^2 = 10$. The pricing problem finds a different extreme ray that also induces an interior direction, indeed $\vec{\boldsymbol{v}}^0 = (\frac{1}{22}, \frac{1}{11}, 0, 0, \frac{-1}{22}, 0, \frac{3}{22})$ that happens to be within the x_1x_2-face.

By no means do we imply that the set $P = \varnothing$ provides all around better directions than with $P = F$. In fact, it suffices to modify the coefficients of x_1 and x_2 in the third constraint to 1 and 3 to get the opposite effect when using $\boldsymbol{w} = \mathbf{1}$. In other words, it remains difficult to distinguish what the quality of a direction really is. For instance, Klee and Minty (1972) propose

Figure 8. Directions $\vec{\boldsymbol{v}}^0$ Found at $\boldsymbol{x}^0$ in Pricing for $P = F$ and Two Variants of $P = \varnothing$

Variant	w_j	c_j coefficients / variables	130 x_1	80 x_2	60 x_3	x_4	x_5	x_6	x_7	z^1	μ_V^0	ρ^0
$P = F$	1	$\vec{\boldsymbol{v}}^0$ (*edge*)	1	0	0	−2	1	−2	1		130	7.5
		$\boldsymbol{x}^1$	7.5	0	0	6	15.5	0	39.5	975		
$P = \varnothing$	1	$\vec{\boldsymbol{v}}^0$ (*interior*)	$\frac{1}{6}$	$\frac{1}{4}$	$\frac{1}{12}$	$\frac{-1}{4}$	0	0	$\frac{1}{4}$		$\frac{140}{3}$	84
		$\boldsymbol{x}^1$	14	21	7	0	8	15	53	3,920		
$P = \varnothing$	$\|\boldsymbol{a}_j\|^2$	$\vec{\boldsymbol{v}}^0$ (*interior/face*)	$\frac{1}{22}$	$\frac{1}{11}$	0	0	$\frac{-1}{22}$	0	$\frac{3}{22}$		$\frac{145}{11}$	176
		$\boldsymbol{x}^1$	8	16	0	21	0	15	56	2,320		

a parametric LP for which the Dantzig's pivot rule behaves exponentially. This is a pathological example not reflecting typical practical behavior but yielding a theoretical insight. Similar examples have been suggested for many known pivoting rules (Paparrizos et al. 2009), and we do so within our framework in the following.

The Klee–Minty polytope is a hypercube of parametric dimension whose corners have been distorted. It can be written as

$$\begin{aligned}\max_{x\geqslant 0}\ & 2^{n-1}x_1 + 2^{n-2}x_2 + \cdots + 2x_{n-1} + x_n \\ \text{s.t.}\ & x_1 \leqslant 5 \\ & 4x_1 + x_2 \leqslant 25 \\ & 8x_1 + 4x_2 + x_3 \leqslant 125 \\ & \vdots \quad \vdots \\ & 2^n x_1 + 2^{n-1}x_2 + \cdots + 4x_{n-1} + x_n \leqslant 5^n.\end{aligned} \tag{29}$$

The previous *LP* contains n variables, n constraints, and 2^n extreme points. Starting at $\boldsymbol{x} = \mathbf{0}$, the initial basis comprises all the slack variables s_j, for $j = 1,\ldots,n$, valued at $s_j = 5^j$. Using the Dantzig's pivot rule, PS first selects the variable with the largest reduced cost (indeed, the largest c_j in this case), that is, x_1 as the entering variable. It then goes through each of the extreme points before reaching the optimal solution at $(0,0,\ldots,5^n)$. None of the $2^n - 1$ pivots is degenerate, hence IPS would follow the same trajectory as PS, this showing that IPS can also have an exponential time complexity in the worst case. MWCC behaves in a totally different way. Assuming unit weights in the pricing problem (21), the latter writes as

$$\begin{aligned}\mu = \max_{y,s\geqslant \mathbf{0}}\ & 2^{n-1}y_1 + 2^{n-2}y_2 + \cdots + 2y_{n-1} + y_n \\ \text{s.t.}\ & y_1 - y_{1+3n} = 0 \\ & 4y_1 + y_2 - y_{2+3n} = 0 \\ & 8y_1 + 4y_2 + y_3 - y_{3+3n} = 0 \\ & \vdots \quad \vdots \\ & 2^n y_1 + 2^{n-1}y_2 + \cdots + 4y_{n-1} + y_n - y_{4n} = 0 \\ & y_1 + y_2 + \cdots + y_{n-1} + y_n + \sum_{j=1}^{n} y_{j+3n} = 1,\end{aligned} \tag{30}$$

where $y_{1+3n}, y_{2+3n}, \ldots, y_{4n}$ are the backward residual variables for the slacks at their upper bounds. Taking into account the last equality to zero together with the normalizing constraint, we derive an upper bound on the value of μ:

$$\begin{aligned}\mu &= \max(2^{n-1}y_1 + 2^{n-2}y_2 + \cdots + 2y_{n-1}) + y_n \\ &= \max\left(\frac{y_{4n} - y_n}{2}\right) + y_n = \max\frac{y_{4n} + y_n}{2} \leqslant \frac{1}{2}.\end{aligned}$$

Let $\mu(y_j) = 2^{n-j}/(2(2^{n-j+1} - 1))$, $j = 1,\ldots,n$, be the value of the objective function in (30) for the cycle using variable y_j, each one inducing an edge direction. The reader can verify that $\mu(y_n) = \frac{1}{2}$, whereas $\mu(y_j) < \frac{1}{2}$ for all $j = 1,\ldots,n-1$. Therefore, MWCC selects the cycle composed of $y_n = 0.5$ and $y_{4n} = 0.5$. The pivot operation raises x_n to 5^n, decreases s_n by the same amount to zero, and therefore MWCC reaches the optimal solution in a single iteration on the Klee–Minty polytope.

4.4. Computational Behavior

Having a generic framework that conceptually hosts, for example, PS, IPS, DCA, and MMCC does not imply that we believe in a generic competitive implementation of the framework. Note that all these specializations are efficiently implemented in the literature, but each implementation massively exploits their respective context. However, our unified presentation opens up further opportunities for more tailoring to particular applications. This may manifest in special kinds of transformation matrices, special update strategies of the row/column partitions, and/or in special strategies of computing dual solutions. We think of this versatility as one of the framework's biggest strengths that may be most useful in contexts in which solving the pricing problem is expensive anyway, like in column generation. An oracle capable of producing strictly improving columns may be extremely useful.

PS at one end of the spectrum is highly competitive, even in network problems where 70%–90% of the pivots are degenerate (Ahuja et al. 1993, Section 18.3). At the other end, MMCC applied on the same type of problems is desperately slow where sometimes solving the oracle takes more time than the original problem (Kovács 2015); we observe the same behavior for the linear programming counterpart MWCC. Nevertheless, when cancel-and-tighten, a clever partial pricing strategy for which the order of the improving cycles becomes irrelevant, is considered alongside MMCC, we see significant time reductions (Gauthier et al. 2015, 2017). In this light, MWCC, just like its network counterpart MMCC, may be an intermediate step to reach a more efficient cancel-and-tighten version adapted to linear programs. This is certainly worth investigating.

Moreover, while IPS requires the inverse of a submatrix within the current basis, the additional computational burden needs to be intimately tied to the implementation for it to make any sense. On that note, it seems that the *positive edge* rule is already implemented in practice (Towhidi et al. 2014).

Finally, DCA is a sound and proven method based on an intuitive clustering of the rows appearing in master problems based on set partitioning formulations. Multiple strategies have been developed to cope with routing and scheduling applications in an efficient manner (Elhallaoui et al. 2008, 2010). While it does not seem

like it can efficiently be applied to arbitrary linear programming matrices, DCA has also been extended to ISUD, which considers integrality in the solution process. The idea is to transfer the integrality requirements to the oracle. The latter aims to determine integer leading directions via ad hoc cutting planes and dynamic weights for the normalization constraint until it fails for lack of tractability. ISUD is as such often able to terminate with a near-optimal solution thus eliminating the typical branch-and-bound tree.

5. Conclusion

This paper unites under one generic framework several known primal algorithms with a broad spectrum of possibilities. Aside from pinpointing reasons simplex-type algorithms suffer from degeneracy, the elimination of the latter is made possible through a linear transformation. The purpose of this paper is further driven by primal algorithms such as column generation where mechanisms coping with degeneracy are beneficial. The two extreme cases of our framework correspond to the primal simplex and the minimum weighted cycle-canceling algorithms. Two properties are established for different family members: positive step sizes at every iteration and pricing problems that provide edge directions only. The improved primal simplex algorithm is remarkably the only variant that qualifies for both features. While interior directions are certainly usual in the realm of nonlinear algorithms, it is not so often that one thinks about such possibilities for simplex-like algorithms. On another note, the minimum weighted cycle-canceling algorithm requires neither a matrix transformation nor the maintenance of basic solutions. It even does not require the knowledge of a dual vector at any iteration, although the convergence proof in strongly polynomial time for the network version is totally driven by the dual point of view. Because the cost and the reduced cost of a cycle are equal, any heuristic looking for negative cycles could interchangeably use the original costs of the variables or some reduced costs derived from any approximation of the dual values. This idea is already proving elegant in our recent experiments with heuristic subproblem pricers within branch and price for vehicle routing problems where columns are found using the original costs rather than the traditional reduced costs.

The vector space decomposition framework revolves around a unique parameter and is derived directly from necessary and sufficient optimality conditions established on the residual problem. The parameter may vary at every iteration and dictates how the linear program decomposition is made. Some variables are hidden, producing an oracle looking for cycles in a projected space where only visible variables remain. Cycles are found and lifted back to the residual problem before finding the step size along the associated direction.

Finally, column generation as a primal algorithm to solve large-scale linear programs is a main beneficiary of our proposal. Indeed, the *dual guided* pricing can reduce the row size of the master problem where degeneracy difficulties occur. Moreover, accelerating strategies such as stabilization techniques can be incorporated to all the variants. Indeed, dual variables can be optimized within intervals in either the master or the subproblem providing flexible arrangements. As we play with dual variables, the automated identification of dual optimal inequalities (Valério de Carvalho 2005, Ben Amor et al. 2006, Gschwind and Irnich 2016) is also appealing.

References

Ahuja RK, Magnanti TL, Orlin JB (1993) *Network Flows: Theory, Algorithms, and Applications* (Prentice Hall, Upper Saddle River, NJ).

Bazaraa MS, Jarvis JJ, Sherali HD (1990) *Linear Programming and Network Flows* (John Wiley & Sons, New York).

Ben Amor HMT, Desrosiers J, Frangioni A (2009) On the choice of explicit stabilizing terms in column generation. *Discrete Appl. Math.* 157(6):1167–1184.

Ben Amor HMT, Desrosiers J, Valério de Carvalho JM (2006) Dual-optimal inequalities for stabilized column generation. *Oper. Res.* 54(3):454–463.

Bixby RE (2002) Solving real-world linear programs: A decade and more of progress. *Oper. Res.* 50(1):3–15.

Dantzig GB, Thapa MN (1997) *Linear Programming* 1*: Introduction*. Mikosch TV, Resnick SI, Robinson SM, eds. Springer Series in Operations Research and Financial Engineering (Springer, New York).

Dantzig GB, Thapa MN (2003) *Linear Programming* 2*: Theory and Extensions*. Mikosch TV, Resnick SI, Robinson SM, eds. Springer Series in Operations Research and Financial Engineering (Springer, New York).

Dantzig GB, Wolfe P (1960) Decomposition principle for linear programs. *Oper. Res.* 8(1):101–111.

du Merle O, Villeneuve D, Desrosiers J, Hansen P (1999) Stabilized column generation. *Discrete Math.* 194(1–3):229–237.

Elhallaoui I, Desaulniers G, Metrane A, Soumis F (2008) Bi-dynamic constraint aggregation and subproblem reduction. *Comput. Oper. Res.* 35(5):1713–1724.

Elhallaoui I, Metrane A, Desaulniers G, Soumis F (2011) An improved primal simplex algorithm for degenerate linear programs. *INFORMS J. Comput.* 23(4):569–577.

Elhallaoui I, Metrane A, Soumis F, Desaulniers G (2010) Multi-phase dynamic constraint aggregation for set partitioning type problems. *Math. Programming* 123(2):345–370.

Elhallaoui I, Villeneuve D, Soumis F, Desaulniers G (2005) Dynamic aggregation of set partitioning constraints in column generation. *Oper. Res.* 53(4):632–645.

Forrest JJ, Goldfarb D (1992) Steepest-edge simplex algorithms for linear programming. *Math. Programming* 57(1):341–374.

Gauthier JB, Desrosiers J, Lübbecke ME (2014) Decomposition theorems for linear programs. *Oper. Res. Lett.* 42(8):553–557.

Gauthier JB, Desrosiers J, Lübbecke ME (2015) About the minimum mean cycle-canceling algorithm. *Discrete Appl. Math.* 196(December):115–134.

Gauthier JB, Desrosiers J, Lübbecke ME (2016) Tools for primal degenerate linear programs: IPS, DCA, and PE. *EURO J. Transportation Logist.* 5(2):161–204.

Gauthier JB, Desrosiers J, Lübbecke ME (2017) A strongly polynomial contraction-expansion algorithm for network flow problems. *Comput. Oper. Res.* 84(August):16–32.

Goldberg AV, Tarjan RE (1989) Finding minimum-cost circulations by canceling negative cycles. *J. ACM* 36(4):873–886.

Gschwind T, Irnich S (2016) Dual inequalities for stabilized column generation revisited. *INFORMS J. Comput.* 28(1):175–194.

Harris PMJ (1973) Pivot selection methods of the Devex LP code. *Math. Programming* 5(1):1–28.

Klee V, Minty GJ (1972) How good is the simplex algorithm? Shisha O, ed. *Inequalities*, Vol. 3 (Academic Press, New York), 159–175.

Kovács P (2015) Minimum-cost flow algorithms: An experimental evaluation. *Optim. Methods Software* 30(1):94–127.

Lübbecke ME, Desrosiers J (2005) Selected topics in column generation. *Oper. Res.* 53(6):1007–1023.

Paparrizos K, Samaras N, Zissopoulos D (2009) Linear programming: Klee–Minty examples. Floudas CA, Pardalos PM, eds. *Encyclopedia of Optimization*, 2nd ed. (Springer, Boston), 1891–1897.

Rosat S, Elhallaoui I, Soumis F, Chakour D (2017a) Influence of the normalization constraint on the integral simplex using decomposition. *Discrete Appl. Math.* 217(January):53–70.

Rosat S, Quesnel F, El Hallaoui I, Soumis F (2017b) Dynamic penalization of fractional directions in the integral simplex using decomposition: Application to aircrew scheduling. *Eur. J. Oper. Res.* 263(3):1007–1018.

Schrijver A (1986) *Theory of Linear and Integer Programming* (John Wiley & Sons, Chichester, West Sussex, UK).

Towhidi M, Desrosiers J, Soumis F (2014) The positive edge criterion within COIN-OR's CLP. *Comput. Oper. Res.* 49(September):41–46.

Valério de Carvalho JM (2005) Using extra dual cuts to accelerate convergence in column generation. *INFORMS J. Comput.* 17(2): 175–182.

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

Jean Bertrand Gauthier is currently at the Johannes Gutenberg University in Mainz, Germany, in a postdoctoral position with Professor Stefan Irnich. His work, "Primal Algorithms for Degenerate Linear and Network Flow Problems," was awarded the best dissertation prize at HEC Montréal (2016).

Jacques Desrosiers is a full professor in the Department of Management Sciences at HEC Montréal. He is also a member of the GERAD Operations Research Center. His main research interests include column generation algorithms and large-scale optimization for vehicle routing and crew scheduling in air, rail, and urban transportation.

Marco E. Lübbecke is a professor and chair of operations research at the School of Business and Economics and the Department of Mathematics at RWTH Aachen University, Germany. He has been modeling and solving large and complex discrete optimization problems from science and industry for more than 20 years. His main research contributions are in computational integer programming, in particular in the field of decomposition algorithms. Marco is an INFORMS board member.