Vector space decomposition for solving large-scale linear programs

Jean Bertrand Gauthier and Jacques Desrosiers
GERAD & HEC Montréal
3000, chemin de la Côte-Sainte-Catherine, Montréal, Canada, H3T 2A7
<jean-bertrand.gauthier, jacques.desrosiers>@hec.ca

Marco E. Lübbecke
RWTH Aachen University
Kackertstraße 7, D-52072 Aachen, Germany
marco.luebbecke@rwth-aachen.de

September 8, 2016

Abstract

This paper describes a vector space decomposition algorithmic framework for linear programming guided by dual feasibility considerations. The resolution process moves from one solution to the next according to an exchange mechanism which is defined by a direction and a post-evaluated step size. The core component 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 upon dividing the set of dual variables in two subsets: one being fixed while the other is optimized. From the primal perspective, one selects a nonnegative combination of variables entering the basis. The direction is uniquely completed by identifying the affected basic variables, if any.

This unified framework is presented in a generic format although it borrows several concepts from network flow problems. Different variants can be extracted several of which corresponding to well known algorithms. The most known special case 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 the so-called minimum weighted cycle-canceling algorithm, a perfect example of network flow analogies introduced in this paper, for which all dual variables are optimized at every iteration. Somewhere in between these two extremes lies another special case, the improved primal simplex algorithm for which one fixes the dual variables associated with the nondegenerate basic variables and optimizes the remaining dual variables. 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.

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 guided by *dual optimality* instead.

While Dantzig's classical pivot rule accurately measures the improvement rate of the objective function, the influence on the (dependent) basic variables is taken for granted for every nonbasic variable unit change. When one ultimately 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. Steepest edge and Devex pivot rules (Forrest and Goldfarb 1992, Harris 1973) alleviate this shortcoming by incorporating in the pricing process additional computations using the basic variable space. In either case, null step sizes are still possible yet can only happen when basic variables are at their 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. We propose the elimination of basic variables leading to degeneracy from consideration in the pricing process which is then sufficient to overcome primal degeneracy. This elimination grants more freedom in the dual variables thus allowing the pricing process flexibility in meeting dual optimality.

We propose a very general algorithm (in fact, a framework) which, given a feasible basic 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, 2008) 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. (2016). 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 re-optimize 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 problem 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 algorithm. 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:

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

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 the matrix $\mathbf{A}$ 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.

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

In Section 2.1, we formulate the so-called *residual problem* which allows the construction of an oracle generating feasible directions in Section 2.2. The latter also provides two alternative primal and dual conditions characterizing optimality for linear programs. Finally, let us embark upon this generic algorithm in Section 2.3 by analyzing a linear transformation, the goal being to structure the technological constraints.

2.1 Residual problem

It is one common practice 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). We do the same with linear programs and define primal-dual optimality conditions on the residual problem.

We define the *residual problem* $LP(\mathbf{x}^k)$ with respect to a given solution $\mathbf{x}^k$ at iteration $k \geq 0$ as follows. Each variable $x_j,\ j \in \{1, \ldots, n\}$, 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 $r_{j+n}^k := x^k - \ell_j$; moreover, only one can be used with a positive value, i.e., the complementarity condition $y_j\, y_{j+n} = 0$ holds, $\forall j \in \{1, \ldots, n\}$, see Figure 1.

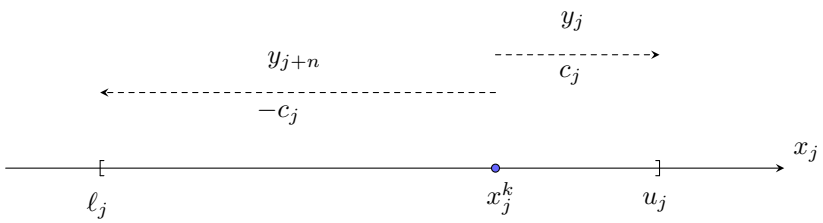

Figure 1: Forward and backward variables for the residual problem

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

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

where $z^k := \mathbf{c}^\intercal \mathbf{x}^k$, $\mathbf{d} := [d_j]_{j\in\{1,\ldots,2n\}}$ is the cost vector, $\mathbf{y} := [y_j]_{j\in\{1,\ldots,2n\}}$ contains the forward and backward variables, their residual upper bounds are given by $\mathbf{r}^k := [r_j^k]_{j\in\{1,\ldots,2n\}}$, and the matrix $\mathbf{K} := [\mathbf{A}, -\mathbf{A}] \equiv [\mathbf{k}_j]_{j\in\{1,\ldots,2n\}}$ stands to remind us that the kernel (or null space) of this matrix is the set of solutions to $\mathbf{K}\mathbf{y} = \mathbf{0}$. Since a variable fixed to 0 is useless, the residual problem $LP(\mathbf{x}^k)$ may be written using only the *residual variables*, that is, the y-variables with positive residual upper bounds within the set $J^k := \{j \in \{1, \ldots, 2n\} \mid r_j^k > 0\}$.

Network flow analogies. Let us propose some linear programming vocabulary in the spirit of network flows. By neglecting the residual upper bounds from (2), one obtains a cone.

Definition 1. *An extreme ray* $\mathbf{y} \in \{\mathbf{K}\mathbf{y} = \mathbf{0}, \mathbf{y} \ge \mathbf{0}\} \subseteq \mathbb{R}^{2n}_+$ *induces a* direction $\vec{\mathbf{v}} \in \mathbb{R}^n$, *not necessarily feasible for LP* (1)*, by computing the differences*

$$\vec{v}_j = y_j - y_{j+n}, \quad \forall j \in \{1, \ldots, n\}. \tag{3}$$

In order to find an improving direction, it then suffices to look within this set of extreme rays. Consider the cone displayed in Figure 2 which illustrates these extreme rays for the solution $\mathbf{x}^k$. 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 a normalization constraint $\mathbf{w}^\intercal \mathbf{y} = 1$, where $\mathbf{w} > \mathbf{0}$ is a vector of arbitrary positive weights associated with the y-variables. This results in a cut cone where every nonnull extreme point corresponds to an extreme ray and thus captures any scaled direction. Furthermore, by considering a different weight vector $\mathbf{w}$, the cutting plane of the cone would be slanted differently thus producing a modified set of extreme points yet each of these would remain associated with the same extreme ray. Definition 2 introduces a normalized extreme ray which can then be manipulated with any scalar.

Definition 2. *Let a* cycle *be a normalized extreme ray* $\mathbf{y}$ *satisfying*

$$\mathbf{y} \in \mathcal{N}^k := \{\mathbf{K}\mathbf{y} = \mathbf{0}, \mathbf{y} \ge \mathbf{0}, \mathbf{w}^\intercal \mathbf{y} = 1\} \subseteq \mathbb{R}^{2n}_+ . \tag{4}$$

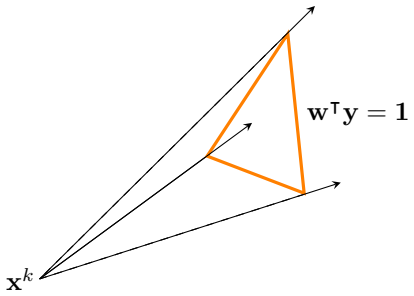

Figure 2: At $\mathbf{x}^k$, the cone $\{\mathbf{y} \geq \mathbf{0} \mid \mathbf{K}\mathbf{y} = \mathbf{0}\} \subseteq \mathbb{R}^{2n}_+$ cut by $\mathbf{w}^\intercal\mathbf{y} = \mathbf{1}$

Because the y-values are unique for a given cycle, it is often simpler to work solely with the variable support denoted by $W := \{j \in \{1,\ldots,2n\} \mid y_j > 0, \mathbf{y} \in \mathcal{N}^k\}$. 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.

Definition 3. *An* augmenting cycle *is a cycle W whose composing y-variables have positive residual upper bounds, that is,*

$$r^k_j > 0,\ \forall j \in W, \quad \textit{or equivalently} \quad W \subseteq J^k\,. \tag{5}$$

When in the following we speak of negative cycles, we always refer to negative *augmenting* cycles. The reduced cost of variable $x_j, j \in \{1,\ldots,n\}$ is defined as $\bar{c}_j := c_j - \boldsymbol{\pi}^\intercal\mathbf{a}_j$ while those of variables y_j and y_{j+n} are respectively

$$\bar{d}_j := c_j - \boldsymbol{\pi}^\intercal\mathbf{a}_j = \bar{c}_j \quad \text{and} \quad \bar{d}_{j+n} := -c_j - \boldsymbol{\pi}^\intercal(-\mathbf{a}_j) = -\bar{c}_j\,. \tag{6}$$

Our cycles share with cycles in networks the property that cost and reduced cost are equal. Indeed, a cycle W satisfies (4), hence $\sum_{j\in W}\mathbf{k}_j y_j = \mathbf{0}$ and

$$\bar{d}_W = \sum_{j\in W}(d_j - \boldsymbol{\pi}^\intercal\mathbf{k}_j)y_j = d_W - \boldsymbol{\pi}^\intercal\sum_{j\in W}\mathbf{k}_j y_j = d_W\,. \tag{7}$$

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

Under the above nomenclature, necessary and sufficient optimality conditions come together in a straightforward manner. In addition to the complementary slackness optimality conditions on LP (1) based on the reduced cost of the original x-variables (see Schrijver 1986), we repeat here two alternative conditions characterizing optimality for linear programs.

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

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

$$\bar{c}_j > 0 \Rightarrow x^k_j = \ell_j; \qquad \bar{c}_j < 0 \Rightarrow x^k_j = u_j; \qquad \ell_j < x^k_j < u_j \Rightarrow \bar{c}_j = 0. \tag{8}$$

Primal: $LP(\mathbf{x}^k)$ *contains no negative cycle, i.e.,*

$$d_W \geq 0, \quad \textit{for every augmenting cycle } W \textit{ in } LP(\mathbf{x}^k). \tag{9}$$

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

$$\bar{d}_j \geq 0, \quad \forall j \in J^k. \tag{10}$$

Assuming the reader accepts complementary slackness conditions, it is worth to sketch the equivalence with the primal and dual conditions. Depending on the value of x^k_j, the residual variables are either y_j (when

$x_j^k = \ell_j$), y_{j+n} (when $x_j^k = u_j$), or both y_j and y_{j+n} (when $\ell_j < x_j^k < u_j$). Recall the signed reduced costs of variables y_j and y_{j+n} in (6) such that all residual variables have a nonnegative reduced cost if and only if complementary slackness is met. With respect to the primal condition claim, when all residual variables have nonnegative reduced costs, then by Lemma 1 $d_W = \bar{d}_W \geq 0$ for every augmenting cycle W in $LP(\mathbf{x}^k)$ and a negative cycle cannot exist.

2.2 Oracle

In order to prove the optimality of $\mathbf{x}^k$ or improve the current solution, we derive an oracle relying on the identification of cycles. One can be derived from the domain (4) and an objective function which effectively computes the cost (or the reduced cost) of each cycle properly as follows

$$\min_{\mathbf{y} \in \mathcal{N}^k} \ \mathbf{d}^\intercal \mathbf{y}. \tag{11}$$

Unfortunately, the oracle (11) may identify non-augmenting cycles. However, it finds an augmenting one (if any) when it uses only residual y-variables, i.e., the y-variables with zero residual capacity are explicitly discarded before removing the upper bounds. Keeping track of the residual variables can be done by partitioning the x-variables according to their current values. In order to achieve this, let $\mathbf{x}^k$ be represented by $(\mathbf{x}_F^k, \mathbf{x}_L^k, \mathbf{x}_U^k)$, where the three sub-vectors respectively refer to the set of free variables $F := \{j \in \{1, \ldots, n\} \mid \ell_j < x_j^k < u_j\}$, at their lower bounds $L := \{j \in \{1, \ldots, n\} \mid x_j^k = \ell_j\}$, and at their upper bounds $U := \{j \in \{1, \ldots, n\} \mid x_j^k = u_j\}$. Let there be $f := |F|$ such free variables, $0 \leq f \leq n$. Observe that if $\mathbf{x}^k$ is basic then $0 \leq f \leq m$. Controlling the presence of the residual variables while solving the oracle can then alternatively be achieved by imposing

$$y_j = 0, \ \forall j \in U \quad \text{and} \quad y_{j+n} = 0, \ \forall j \in L. \tag{12}$$

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 $\rho\, y_j \leq r_j^k, \ \forall j \in W$ and this cycle is *canceled* when the step size is equal to

$$\rho := \min_{j \in W} \ \frac{r_j^k}{y_j} > 0. \tag{13}$$

Primal simplex algorithm. Consider a basic solution $\mathbf{x}^k$ 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 $x_\ell, \ \ell \in N$. 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 maximizes the exchange potential of the entering variable and it maintains a basic feasible solution for $\mathbf{x}^{k+1}$. 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 $k \geq 0$, it goes from vertex $\mathbf{x}^k$ represented by a nonoptimal basis to vertex $\mathbf{x}^{k+1}$ 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 $\vec{\mathbf{v}}$ (see Definition 1)

$$\vec{v}_j = \begin{cases} y_j - y_{j+n}, & \forall j \in B \cup \{\ell\} \\ 0, & \forall j \in N \setminus \{\ell\} \end{cases} \tag{14}$$

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 $\mathbf{a}_\ell$ of the entering variable as part of the linear span of $\mathbf{A}_B$, that is, $\mathbf{V}(\mathbf{A}_B) = \mathbb{R}^m$. By definition, any m-dimensional column belongs to $\mathbf{V}(\mathbf{A}_B)$ meaning in particular that for any nonbasic entering variable

$$\exists! \boldsymbol{\lambda} \in \mathbb{R}^m \text{ such that } \sum_{j \in B} \mathbf{a}_j \lambda_j = \mathbf{a}_\ell \text{ which works out to } \boldsymbol{\lambda} = \mathbf{A}_B^{-1} \mathbf{a}_\ell \,. \tag{15}$$

Observe from (14) 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 basic ones. Since the linear combination scalars $\boldsymbol{\lambda}$ in (15) can take any sign, every column of $\mathbf{A}_B$ 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 variable is defined by a linear combination containing at least one degenerate variable, that is, a basic variable at one of its bounds. Indeed, the associated cycle to such an entering variable *might* include a y_j-variable, $j \in B$, with a residual upper bound of 0, i.e., a forward variable $y_j > 0 \mid x_j^k = u_j$ or a backward variable $y_{j+n} > 0 \mid x_j^k = \ell_j$. The reader may want to compare this with (12) to realize that the degeneracy phenomenon takes an equivalent form in the oracle as well.

Recall that the presence of degeneracy in a basic solution $\mathbf{x}^k$ to LP (1) is detected when only a strict subset F of the basic variables are free, say $\emptyset \subseteq F \subset B$. As degenerate pivots can only occur when the current solution is degenerate, the following section derives a so-called *transformation matrix* which capitalizes on the over representation notion attached to degenerate solutions.

2.3 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 $\mathbf{T}_P^{-1}$ we propose in (17) is to structure the technological constraints, where $P \subseteq B$ is an ordered subset of the indexes of basic variables. Set P induces a *subspace basis* $\mathbf{A}_P$ with dimension $p := |P|$. In that case, a subset of p rows within $\mathbf{A}_P$ are independent. Then there exists a row partition $\left[\begin{smallmatrix} \mathbf{A}_{RP} \\ \mathbf{A}_{SP} \end{smallmatrix}\right]$ of $\mathbf{A}_P$ such that $\mathbf{A}_{RP}$ is a nonsingular matrix of size $p \times p$. For instance, an optimal basic solution to the *restricted* phase I problem

$$\min\{\mathbf{1}^\intercal \boldsymbol{\theta} \mid \mathbf{A}_P \mathbf{x}_P + \mathbf{I}_m \boldsymbol{\theta} = \mathbf{b}, \boldsymbol{\theta} \geq \mathbf{0}\} \tag{16}$$

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

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

provides basis $\mathbf{T}_P$ of $\mathbb{R}^m$ and its inverse of particularly simple structure:

$$\mathbf{T}_P = \begin{bmatrix} \mathbf{I}_p & \mathbf{0} \\ \mathbf{M} & \mathbf{I}_{m-p} \end{bmatrix} \quad \text{and} \quad \mathbf{T}_P^{-1} = \begin{bmatrix} \mathbf{I}_p & \mathbf{0} \\ -\mathbf{M} & \mathbf{I}_{m-p} \end{bmatrix}. \tag{17}$$

Applying the linear transformation $\mathbf{T}_P^{-1}$ on the system $\mathbf{Ax} = \mathbf{b}$ results in $\bar{\mathbf{A}} := \mathbf{T}_P^{-1}\mathbf{A}$ and $\bar{\mathbf{b}} := \mathbf{T}_P^{-1}\mathbf{b}$ as follows:

$$\bar{\mathbf{A}} = \begin{bmatrix} \mathbf{A}_R \\ \mathbf{A}_S - \mathbf{M}\mathbf{A}_R \end{bmatrix} \quad \text{and} \quad \bar{\mathbf{b}} = \begin{bmatrix} \mathbf{b}_R \\ \mathbf{b}_S - \mathbf{M}\mathbf{b}_R \end{bmatrix}. \tag{18}$$

The point of the row partition all comes together in the following definition by reminiscing on the outcome of Gauss-Jordan elimination on linear dependent systems.

Definition 4. (Gauthier et al. 2015b, Proposition 3) *A vector* $\mathbf{a} \in \mathbb{R}^m$ *(and the associated variable, if any) is* compatible with $\mathbf{A}_P$ *if and only if* $\bar{\mathbf{a}}_S := \mathbf{a}_S - \mathbf{M}\mathbf{a}_R = \mathbf{0}$ *or, equivalently, it belongs to* $\mathbf{V}(\mathbf{A}_P)$*, the linear span of the vector subspace basis* $\mathbf{A}_P$*. Otherwise, the vector* $\mathbf{a}$ *is* incompatible.

Verifying the equivalence is straightforward when decomposing $\begin{bmatrix} \mathbf{a}_R \\ \mathbf{a}_S \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{RP} \\ \mathbf{A}_{SP} \end{bmatrix} \boldsymbol{\lambda}$. 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 matrix $\mathbf{A}$, the most probing known cases being the network and set partitioning problems which easily permit the verification of the definition, see the *Transformation matrix insight* paragraph at the end of this section. Compatibility can also be determined using the *positive edge* rule (see Towhidi et al. 2014) which uses a stochastic argument to reduce the matrix multiplication computational penalty, and testing for compatibility over basis $\mathbf{A}_B$ can be done in $O(m^2)$.

Let $Q := \{1, \ldots, n\} \setminus P$ contain all the variables outside the set P. This column partition is represented by the matrix $\mathbf{A} = \begin{bmatrix} \mathbf{A}_P, \mathbf{A}_Q \end{bmatrix}$. Altogether, we have the row/column partition

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{RP} & \mathbf{A}_{RQ} \\ \mathbf{A}_{SP} & \mathbf{A}_{SQ} \end{bmatrix},$$

where the nonsingular $p \times p$ matrix $\mathbf{A}_{RP}$ is called the *working basis*. Applying $\mathbf{T}_P^{-1}$ on $\mathbf{A}$ yields

$$\bar{\mathbf{A}} = \begin{bmatrix} \mathbf{I}_p & \mathbf{0} \\ -\mathbf{M} & \mathbf{I}_{m-p} \end{bmatrix} \begin{bmatrix} \mathbf{A}_{RP} & \mathbf{A}_{RQ} \\ \mathbf{A}_{SP} & \mathbf{A}_{SQ} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{RP} & \mathbf{A}_{RQ} \\ \mathbf{0} & \bar{\mathbf{A}}_{SQ} \end{bmatrix}. \tag{19}$$

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

Transformation matrix insight. Ultimately, the transformation matrix produces row and column partitions intimately binded 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 $\mathbf{A}_F$ can be separated in trees forming a forest. The latter is expressed in matrix form in Figure 3a 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 $\mathbf{A}_F$ thus forming the row partition presented in Figure 3b. A constructive approach leading to a contracted network for the identification of negative cycles is presented in Gauthier et al. (2016).

	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							

(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				
R	7				1			1
	8				-1	-1	1	
	9					1		
	10						-1	
	1			**1**				
S	2							**-1**
	3							

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

Figure 3: Network flow problem row partition

In set partitioning problems, and more specifically when using DCA, a subspace basis $\left[\begin{smallmatrix}\mathbf{I}_p\\ \mathbf{M}\end{smallmatrix}\right]$ is obtained on-the-fly while *heuristically* trying to select linear independent rows within $\mathbf{A}_F$. This process is sketched in Figure 4. In Figure 4a, the original matrix $\mathbf{A}_F$ is presented whereas Figure 4b 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 4b obtains five rows in the set R and three in the set S. Figure 4c uses these identifiers by replacing the matrix content with trivial unit references for each identifier, thus obtaining the subspace basis $\left[\begin{smallmatrix}\mathbf{I}_5\\ \mathbf{M}\end{smallmatrix}\right]$. One can easily verify that the four columns of $\mathbf{A}_F$ in Figure 4b belong to the span of that subspace basis. This is also true for a fifth column from the simplex basis $\mathbf{A}_B$, its actual content being irrelevant.

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

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

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

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

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

(c) Subspace basis $\left[\begin{smallmatrix}\mathbf{I}_5\\ \mathbf{M}\end{smallmatrix}\right]$

Figure 4: Set partitioning problem row partition

3 Vector space decomposition framework

In this section, we look at the essential components of the framework we propose. The algorithm relies on an oracle to iterate. The latter is dynamically updated with respect to the values of the current solution $\mathbf{x}^k$. As such, we already stated the residual problem which gives a great deal of insight to the oracle's solutions. We also define a row/column partition, based on the transformation matrix, that generates the content of the oracle. 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 algorithm is broken down into nine main steps aside from the initialization. Figure 5 provides an overview of these steps, while the following subsections detail their content.

Initialization: Iteration $k := 0$;
Feasible solution $\mathbf{x}^0$;
Select the subspace basis $\mathbf{A}_P$;
1 Derive the transformation matrix $\mathbf{T}_P^{-1}$ and the row/column partitions $\{R, S\}/\{P, Q\}$ of $\mathbf{A}$;
2 Divide the vector $\mathbf{y} \in \mathbb{R}^{2n}_+$ with the three sets H_P, V_P and Z_P as in (20)–(22);
3 Solve the pricing problem (27) for μ_V^k and the projected cycle W_V^k;
4 If $\mu_V^k \geq 0$, **terminate** with an optimality certificate for $\mathbf{x}^k$;
5 Extract the lifted cycle $W_{V\cup H}^k$ from (28)–(29) and the direction $\vec{\mathbf{v}}^k$;
6 Compute the step size ρ^k (30);
7 Update to solution $\mathbf{x}^{k+1} := \mathbf{x}^k + \rho^k \vec{\mathbf{v}}^k$ using the system (31);
8 Update the subspace basis $\mathbf{A}_P$;
9 Iterate $k := k + 1$;

Figure 5: Generic vector space decomposition algorithm for linear programs

Initialization. It all starts at iteration $k = 0$ with some basic feasible solution $\mathbf{x}^0$ and column partition $\{F, L, U\}$. The construction of the residual problem $LP(\mathbf{x}^k)$ (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 $0 \leq y_j \leq r_j^k,\ \forall j \in \{1, \ldots, 2n\}$.

3.1 Structured residual problem

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

$$H_P \equiv H_P(\mathbf{x}^k) := \bigcup_{j \in P} \{j, j + n\} \tag{20}$$

$$V_P \equiv V_P(\mathbf{x}^k) \ := J^k \setminus H_P \tag{21}$$

$$Z_P \equiv Z_P(\mathbf{x}^k) \ := \{1, \ldots, 2n\} \setminus \{H_P \cup V_P\}\,, \tag{22}$$

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 $j \in U$, the variable 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

$$\begin{array}{rlcccl} z^\star = z^k + & \min & \displaystyle\sum_{j\in H_P} d_j y_j & + & \displaystyle\sum_{j\in V_P} d_j y_j & \\ & \text{s.t.} & \displaystyle\sum_{j\in H_P} \mathbf{k}_{Rj} y_j & + & \displaystyle\sum_{j\in V_P} \mathbf{k}_{Rj} y_j & = \mathbf{0} \quad [\boldsymbol{\psi}_R] \\ & & & & \displaystyle\sum_{j\in V_P} \bar{\mathbf{k}}_{Sj} y_j & = \mathbf{0} \quad [\boldsymbol{\psi}_S] \\ & 0 \le y_j \le r_j^k,\ \forall j \in H_P, & & & 0 \le y_j \le r_j^k,\ \forall j \in V_P, & \end{array} \tag{23}$$

where $\mathbf{k}_j$ is the j-th column vector of $\mathbf{K}$, $\bar{\mathbf{K}} := \mathbf{T}_P^{-1}\mathbf{K}$, 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 $\boldsymbol{\pi}^\intercal = \boldsymbol{\psi}^\intercal \mathbf{T}_P^{-1}$, that is, $[\boldsymbol{\pi}_R^\intercal,\ \boldsymbol{\pi}_S^\intercal] = [\boldsymbol{\psi}_R^\intercal - \boldsymbol{\psi}_S^\intercal \mathbf{M},\ \boldsymbol{\psi}_S^\intercal]$.

3.2 Pricing: cycle in $\mathbb{R}^{2n}_+$ and direction in $\mathbb{R}^n$

The pricing problem exploits the structure in (23) 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 which 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]:

$$\mathbf{0} = \bar{\mathbf{c}}_P^\intercal = \mathbf{c}_P^\intercal - \boldsymbol{\psi}_R^\intercal \mathbf{A}_{RP} \quad (= \mathbf{c}_P^\intercal - (\boldsymbol{\pi}_R^\intercal + \boldsymbol{\pi}_S^\intercal \mathbf{M})\mathbf{A}_{RP} = \mathbf{c}_P^\intercal - \boldsymbol{\pi}_R^\intercal \mathbf{A}_{RP} - \boldsymbol{\pi}_S^\intercal \mathbf{A}_{SP}). \tag{24}$$

This is equivalent to imposing $\bar{d}_j = d_j - \boldsymbol{\psi}_R^\intercal \mathbf{k}_{Rj} \ge 0,\ \forall j \in H_P$, that is, $\bar{d}_j = \bar{d}_{j+n} = 0,\ \forall j \in P$. Furthermore, if the current solution $\mathbf{x}^k$ is also optimal, there should exist 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]. This verification can be done with the linear program

$$\begin{array}{l} \max\ \mu_V \\ \quad \text{s.t. } \mu_V \le \bar{d}_j = d_j - \boldsymbol{\psi}_R^\intercal \mathbf{k}_{Rj} - \boldsymbol{\psi}_S^\intercal \bar{\mathbf{k}}_{Sj} \quad [y_j] \quad \forall j \in V_P, \end{array} \tag{25}$$

where the vector $\boldsymbol{\psi}_R^\intercal = \mathbf{c}_P^\intercal \mathbf{A}_{RP}^{-1}$ is fixed by (24) 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 the *Oracle interpretation* paragraph at the end of this section):

$$\begin{array}{l} \max\ \mu_V \\ \quad \text{s.t. } \mu_V \le \dfrac{\bar{d}_j}{w_j} = \dfrac{1}{w_j}(d_j - \boldsymbol{\psi}_R^\intercal \mathbf{k}_{Rj} - \boldsymbol{\psi}_S^\intercal \bar{\mathbf{k}}_{Sj}) \quad [y_j] \quad \forall j \in V_P. \end{array} \tag{26}$$

Dualizing (26), we obtain the primal form of the oracle which comprises $m-p+1$ constraints and writes as

$$\begin{aligned} \min \quad & \sum_{j\in V_P} \tilde{d}_j y_j \\ \text{s.t.} \quad & \sum_{j\in V_P} \bar{\mathbf{k}}_{Sj} y_j = \mathbf{0} && [\boldsymbol{\psi}_S] \\ & \sum_{j\in V_P} w_j\, y_j = 1 && [\mu_V] \\ & y_j \geq 0, \quad \forall j \in V_P, \end{aligned} \tag{27}$$

where $\tilde{d}_j := d_j - \boldsymbol{\psi}_R^\intercal \mathbf{k}_{Rj} = d_j - \mathbf{c}_P^\intercal \mathbf{A}_{RP}^{-1}\mathbf{k}_{Rj},\ \forall j \in V_P$. The oracle interpretation is done through the primal/dual pair (26)/(27). It brings together the negative cycles and the transformation matrix $\mathbf{T}_P^{-1}$.

Oracle interpretation. First of all, the formulation (27) is always feasible unless $\mathbf{x}^0$ is the only feasible solution of LP (1). Reciprocally, the formulation (26) 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 $\mu_V \leq 0$. Furthermore, weight values of $\mathbf{w}$ used for the normalization constraint can be set in stone or updated dynamically. In the former case, think of the simple one vector typically used in network flows (see Gauthier et al. 2015a) or the norm based weights such as $w_j = w_{j+n} = \|\mathbf{a}_j\|,\ \forall j \in \{1,\ldots,n\}$, 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 towards or away from certain solutions, see (Rosat et al. 2016a,b) for several alternatives which have been particularly successful for solving set partitioning problems using the *integral simplex using decomposition* algorithm (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 $\mathbf{y}_V^k := [y_j^k]_{j\in V_P}$ denote an optimal solution to the primal/dual system of value μ_V^k at iteration $k \geq 0$. If $\mu_V^k \geq 0$, the dual optimality conditions are satisfied and the algorithm terminates with an optimal solution $\mathbf{x}^k$. Otherwise, $\mu_V^k < 0$ and the current solution might be *improved* by following a direction.

An optimal negative cycle, say W_V^k, derived from (27) is augmenting since only residual variables $y_j, j \in V_P$, are considered (see Definition 3). However, the fact that $\mathbf{y}_V^k$ is built omitting variables $\mathbf{y}_H$ means that it only provides a *portion* of $\mathbf{y}^k$. Then again, by construction of the linear transformation, these hidden components are uniquely determined within (23) in the system of row set R where the free nature of variables in P is assumed, that is,

$$\begin{aligned} & \sum_{j\in H_P} \mathbf{k}_{Rj} y_j \quad + \quad \sum_{j\in V_P} \mathbf{k}_{Rj} y_j^k = \mathbf{0} \\ & y_j \geq 0,\ \forall j \in H_P\,. \end{aligned} \tag{28}$$

The solution $\mathbf{y}_H^k$ of this system is determined alike (15) by $\boldsymbol{\lambda}_P := \mathbf{A}_{RP}^{-1}\left(-\sum_{j\in V_P} \mathbf{k}_{Rj} y_j^k\right)$ as

$$y_j^k = \begin{cases} -\lambda_j, & \text{if } \lambda_j < 0 \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad y_{j+n}^k = \begin{cases} \lambda_j, & \text{if } \lambda_j > 0 \\ 0, & \text{otherwise} \end{cases} \quad \text{for all } j \in P\,, \tag{29}$$

and the direction $\vec{\mathbf{v}}^k$ follows from the application of Definition 1 to $\mathbf{y}^k = [\mathbf{y}_H^k,\ \mathbf{y}_V^k,\ \mathbf{y}_Z^k]$. 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, $\mathbf{y}_H^k$ is established in (29) 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 (27) whereas the full cycle produced with $[\mathbf{y}_H^k,\ \mathbf{y}_V^k]$ 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 cost of the hidden component 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).

3.3 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

$$\rho^k := \min_{j \in W_{V\cup H}^k} \left\{ \frac{r_j^k}{y_j^k} \right\} \geq 0\,. \tag{30}$$

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

$$\forall j \in \{1, \ldots, n\}, \qquad x_j^{k+1} := \begin{cases} x_j^k + \rho^k\, y_j^k\,, & \text{if } j \in W_{V\cup H}^k \\ x_j^k - \rho^k\, y_j^k\,, & \text{if } j+n \in W_{V\cup H}^k \\ x_j^k, & \text{otherwise.} \end{cases} \tag{31}$$

Depending on the choice of the subspace basis $\mathbf{A}_P$, $\mathbf{x}^{k+1}$ represented by $[\mathbf{x}_F^{k+1}, \mathbf{x}_L^{k+1}, \mathbf{x}_U^{k+1}]$ 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 $\mathbf{x}^{k+1}$ can be rendered basic by solving a restricted problem over the set of free variables:

$$\begin{aligned} z^{k+1} = \min \quad & \mathbf{c}_F^\intercal \mathbf{x}_F \;+\; \mathbf{c}_L^\intercal \mathbf{x}_L \;+\; \mathbf{c}_U^\intercal \mathbf{x}_U \\ \text{s.t.} \quad & \mathbf{A}_F \mathbf{x}_F \;+\; \mathbf{A}_L \mathbf{x}_L \;+\; \mathbf{A}_U \mathbf{x}_U = \mathbf{b} \\ & \mathbf{l}_F \leq \mathbf{x}_F \leq \mathbf{u}_F, \quad \mathbf{x}_L = \mathbf{x}_L^{k+1}, \quad \mathbf{x}_U = \mathbf{x}_U^{k+1}. \end{aligned} \tag{32}$$

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

4 Properties

The generic algorithm 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 generic framework whereas Section 4.2 qualifies the kinds of directions found with the pricing problem 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.

4.1 Special cases

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

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

Proof. If $P \subseteq F$, or equivalently $P \cap \{L \cup U\} = \emptyset$, then all y-variables in H_P are residual as well as those in V_P. Therefore, the primal/dual pair of the pricing problem (26)/(27), respectively, match 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 $LP(\mathbf{x}^k)$, meaning that the associated step size is positive. □

Remark. Consider the case $P \nsubseteq F$, or equivalently $P \cap \{L \cup U\} \neq \emptyset$, 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 $\boldsymbol{\psi}_R$. 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, i.e., 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 $P = \emptyset$. When choosing $P = \emptyset$, it amounts to a subspace basis $\mathbf{A}_\emptyset$ of dimension zero which in turn means that $V(\mathbf{A}_\emptyset) = \{\mathbf{0}\}$. Since the vector subspace contains only the null vector, there are no compatible variables, basic or otherwise. The linear transformation is trivial, that is, $\mathbf{T}_\emptyset = \mathbf{T}_\emptyset^{-1} = \begin{bmatrix} \mathbf{I}_0 & \emptyset \\ \emptyset & \mathbf{I}_m \end{bmatrix} = \mathbf{I}_m$. 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 $\mathbf{A}$ is the node-arc incidence matrix of a network, this particular case corresponds to the remarkable strongly polynomial minimum mean cycle-canceling algorithm 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 of MMCC are also transferable in the latter is left for another paper.

Case $P = F$. When choosing $P = F$, it corresponds to the strategy developed by Elhallaoui et al. (2011) in IPS. The vector subspace $\mathbf{V}(\mathbf{A}_F)$ includes all columns of $\mathbf{A}_F = \begin{bmatrix} \mathbf{A}_{RF} \\ \mathbf{A}_{SF} \end{bmatrix}$ but none associated with the degenerate basic variables: $\forall j \in B,\ \mathbf{a}_j \in \mathbf{V}(\mathbf{A}_F) \Leftrightarrow j \in F$. The linear transformation is given by $\mathbf{T}_F^{-1} = \begin{bmatrix} \mathbf{I}_f & \mathbf{0} \\ -\mathbf{M} & \mathbf{I}_{m-f} \end{bmatrix}$, where $\mathbf{M} = \mathbf{A}_{SF}\mathbf{A}_{RF}^{-1}$. 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 $\mathbf{A}_B$ and $\mathbf{V}(\mathbf{A}_B) = \mathbb{R}^m$. All variables are compatible whereas the sets P and Q, respectively, correspond to all basic and nonbasic variables. The subspace basis induces $\mathbf{T}_B = \mathbf{T}_B^{-1} = \left[\begin{smallmatrix} \mathbf{I}_m & \emptyset \\ \emptyset & \mathbf{I}_0 \end{smallmatrix}\right] = \mathbf{I}_m$ yielding once again a trivial transformation. Most importantly, it fixes $\boldsymbol{\pi}^\intercal = \mathbf{c}_B^\intercal \mathbf{A}_B^{-1}$, that is, *all* dual variables, and with $\mathbf{w} = \mathbf{1}$, the generic algorithm becomes the primal simplex with Dantzig's pivot-selection rule. When $B \cap \{L \cup U\} \neq \emptyset$, 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, i.e., a degenerate pivot.

Case $P \supseteq F$. When choosing $P \supseteq F$, we have $f \leq p \leq m$ linearly independent column vectors in $\mathbf{A}_P$. This is the strategy used in Elhallaoui et al. (2005, 2008) for solving set partitioning problems by column generation. While the equivalent form with the columns of $\mathbf{A}_B$ exists, the subspace basis $\left[\begin{smallmatrix} \mathbf{I} \\ \mathbf{M} \end{smallmatrix}\right]$ 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. Indeed, in typical applications, a vehicle route or a crew schedule 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 $m/\bar{m}$. DCA thus capitalizes on the characterization of set partitioning optimal binary solutions which are usually highly degenerate. The compatibility interpretation follows by design, a column $\mathbf{a}_j$, $j \in \{1, \ldots, n\}$, of $\mathbf{A}$ is compatible with the row clustering if and only if the itinerary or schedule respects that clustering into multi-task activities.

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

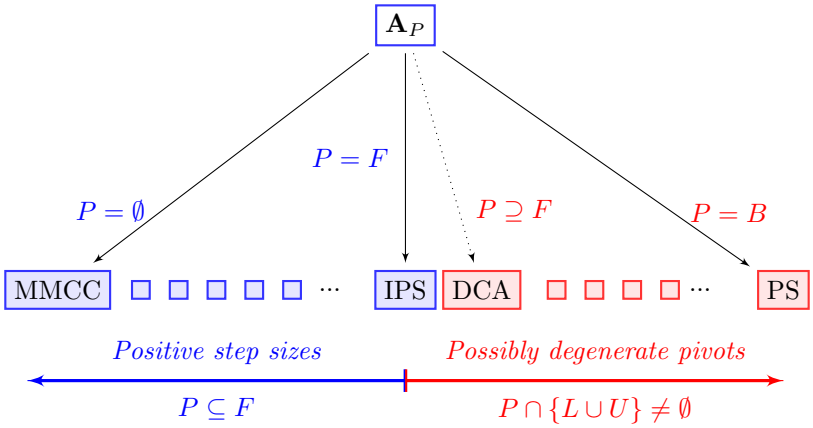

Figure 6: Special cases of VSD: MMCC (1989), IPS (2008), DCA (2005), and PS (1947)

4.2 Interior directions

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

An important property of the IPS algorithm is its movement on an edge of the polyhedron defined by the set of constraints of LP (1) at every iteration. In essence, the proof is as follows. Consider a generic basis

composed of the columns of $\mathbf{A}_F$ completed with $m-f$ artificial variables at zero. Because the index set of the visible y-variables is $V_P^k = L \cup U$ at iteration k, either a single compatible variable or a combination of at most $m-f+1$ incompatible variables at their lower or upper bounds is selected in the cycle W_V^k. In the former case, it acts as in PS with a nondegenerate pivot, hence a movement along an edge. In the latter case, the selected incompatible variables can enter the current basis one by one with degenerate pivots, each pivot removing an artificial variable thus maintaining the basis status of the solution, whereas the last pivot is nondegenerate, hence the movement is made along an edge. Let us formalize this result.

Proposition 3. (Elhallaoui et al. 2011, Proposition 4) *Let $\mathbf{x}^k$, $k \geq 0$, be a nonoptimal basic solution to LP (1). For $P = F$, the direction $\vec{\mathbf{v}}^k$ is an edge direction.*

Proposition 2 shows that the family of algorithms with $P \cap \{L \cup U\} = \emptyset$, 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 $\mathbf{x}^k$. In a nut shell, since the case $P = F$ only identifies edge directions by Proposition 3, it provides the smallest dimensional cone $\bar{K}_{SV}\mathbf{y}_V = \mathbf{0}, \mathbf{y} \geq \mathbf{0}$, 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, i.e., interior directions.

Proposition 4. *Let $\mathbf{x}^k$, $k \geq 0$, be a nonoptimal basic solution to LP (1). For $P \subset F$, an interior direction $\vec{\mathbf{v}}^k$, if any, can be expressed as a nonnegative combination of the edge directions.*

Proof. For $P = F$, any lifted solution $\mathbf{y}^k = [\mathbf{y}_V^k, \mathbf{y}_H^k, \mathbf{y}_Z^k]$ to (27)–(29) 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 $\mathbf{y}$ to (27)–(29) is nonnull and by the representation theorems of Minkowski and Weyl (see Schrijver 1986), it can then be expressed as a nonnegative combination of these extreme rays, that is,

$$\mathbf{y} = \sum_{\omega \in \Omega_F^k} \mathbf{y}^\omega \lambda^\omega, \qquad \lambda^\omega \geq 0, \ \forall \omega \in \Omega_F^k, \tag{33}$$

or component-wise for $j \in \{1, \ldots, 2n\}$, $y_j = \sum_{\omega \in \Omega_F^k} y_j^\omega \lambda^\omega, \lambda^\omega \geq 0, \ \forall \omega \in \Omega_F^k$. By Definition 1, every component $\vec{v}_j, j \in \{1, \ldots, n\}$, of the associated direction $\vec{\mathbf{v}}$ is given by

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

and hence, any direction $\vec{\mathbf{v}}$ can be expressed as a nonnegative combinations of the edge directions:

$$\vec{\mathbf{v}} = \sum_{\omega \in \Omega_F^k} \mathbf{v}^\omega \lambda^\omega, \qquad \lambda^\omega \geq 0, \ \forall \omega \in \Omega_F^k. \tag{34}$$

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 $\vec{\mathbf{v}}^k$ is not an edge direction, then it is interior and by (34) 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 $P = \emptyset$ is one of the variants susceptible to lead to nonbasic solutions. However, since $\mathbf{T}_\emptyset^{-1} = \mathbf{I}_m$ and all dual variables are optimized in the pricing problem, the simplifications applicable to the different

steps of the generic algorithm 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 (32).

4.3 Illustrative example

Consider the linear program expressed in (35) 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 $\mathbf{x}^0$ 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 and hence there are three edge directions according to the selected entering variable x_1, x_2, or x_3.

$$\begin{array}{rrrrl} \max & 130x_1 & +80x_2 & +60x_3 & \\ \text{s.t.} & 2x_1 & - x_2 & + 2x_3 & \leq 21 \\ & - x_1 & + x_2 & - x_3 & \leq 8 \\ & 2x_1 & - x_2 & - x_3 & \leq 15 \\ & - x_1 & - x_2 & + 2x_3 & \leq 32 \\ & x_1, & x_2, & x_3 & \geq 0 \end{array} \tag{35}$$

Let us start with the possible values of the projected vector $\mathbf{y}_V \in \mathbb{R}_+^{|V|}$ arising from $\mathbf{y} \in \mathbb{R}_+^{14}$ while solving the oracle (27). 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 (27), 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 $P = \emptyset$, precisely all the residual variables are visible in the oracle, none are hidden as $H_P = \emptyset$, and y_8, y_9 and y_{10} are again fixed to zero. The vector $\mathbf{y}_V$ 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 integer numbers, are listed in Figure 7.

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											
$P = \emptyset$	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

Figure 7: Extreme rays $\mathbf{y}_V$ at $\mathbf{x}^0$ in pricing for $P = F$ and $P = \emptyset$

Figure 8 details the content of the direction $\vec{\mathbf{v}}^0 \in \mathbb{R}^7$ found under the suggested set P and weight vector $\mathbf{w}$. The minimum cost of the associated cycle W_V^0 is given by μ_V^0 and the step size ρ^0 can be recovered with (30). Finally, the new solution $\mathbf{x}^1$ is obtained with (31). As expected, the direction $\vec{\mathbf{v}}^0 = (1, 0, 0, -2, 1, -2, 1)$ found with the improved primal simplex algorithm ($P = F$) follows an edge and yields an extreme point solution at $\mathbf{x}^1$. Notice that since the current solution $\mathbf{x}^0$ is nondegenerate, we have $F = B$ and we would have found the same direction upon selecting x_1 as the entering variable in the primal simplex algorithm.

However, when $P = \emptyset$ and $w_j = 1$, $\forall j \in \{1, \ldots, 14\}$, the direction $\vec{\mathbf{v}}^0 = (1/6, 1/4, 1/12, -1/4, 0, 0, 1/4)$ induced by the last extreme ray of Figure 7 so happens to be interior (see Figure 9) and yields a feasible solution $\mathbf{x}^1$ which is nonbasic (six variables take positive values). By Proposition 4, this $\vec{\mathbf{v}}^0$ is indeed the combination of

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	$\vec{\mathbf{v}}^{\,0}$ (**edge**)	1	0	0	-2	1	-2	1		130	7.5
		$\mathbf{x}^1$	7.5	0	0	6	15.5	0	39.5	975		
$P = \emptyset$	1	$\vec{\mathbf{v}}^{\,0}$ (**interior**)	1/6	1/4	1/12	-1/4	0	0	1/4		140/3	84
		$\mathbf{x}^1$	14	21	7	0	8	15	53	3920		
$P = \emptyset$	$\|\mathbf{a}_j\|^2$	$\vec{\mathbf{v}}^{\,0}$ (**interior/face**)	1/22	1/11	0	0	-1/22	0	3/22		145/11	176
		$\mathbf{x}^1$	8	16	0	21	0	15	56	2320		

Figure 8: Directions $\vec{\mathbf{v}}^{\,0}$ found at $\mathbf{x}^0$ in pricing for $P = F$ and two variants of $P = \emptyset$

edge directions, that is,

$$\begin{aligned}(1/6, 1/4, 1/12, -1/4, 0, 0, 1/4) \;&=\; 1/6\;\; (1,0,0,-2,1,-2,1)\\ &+\; 1/4\;\; (0,1,0,1,-1,1,1)\\ &+\; 1/12\; (0,0,1,-2,1,1,-2).\end{aligned}$$

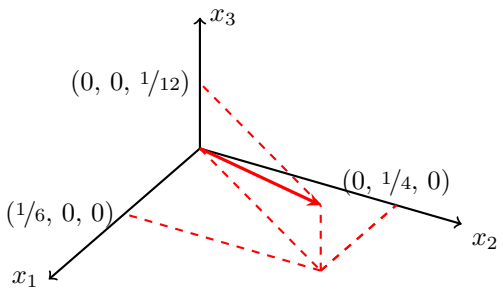

Figure 9: Three-dimensional interior direction $(\vec{v}_1^{\,0}, \vec{v}_2^{\,0}, \vec{v}_3^{\,0}) = (1/6, 1/4, 1/12)$ with $P = \emptyset$ and $\mathbf{w} = \mathbf{1}$

By no means do we imply that the set $P = \emptyset$ 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 the uniform one weight vector. The last example in Figure 8 still uses the set $P = \emptyset$ but uses a weight vector whose every element w_j is determined by computing the squared norm $\|\mathbf{a}_j\|^2$ of each column, i.e., $w_1 = 2^2 + (-1)^2 + 2^2 + (-1)^2 = 10$. The pricing problem finds a different extreme ray which also induces an interior direction, indeed $\vec{\mathbf{v}}^{\,0} = (1/22, 1/11, 0, 0, -1/22, 0, 3/22)$ that happens to be within the x_1x_2-face.

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 which 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. In our framework, the oracle for such findings does not require derivatives and

in fact remains linear. On another note, the minimum weighted cycle-canceling algorithm requires neither matrix transformation nor the maintenance of basic solutions.

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.

The implementation of this generic framework is a chapter of its own. While this paper has no computational study pretension, several ideas have already been suggested as promising assets. The transformation matrix induces structure in the technological matrix whereas the residual problem automates degeneracy screening. By combining both constructions, extensive variable screening in the pricing problem can be done on several fronts such as compatibility and partial reduced costs to residual upper bounds ratios. The first kind being easy performed pivots whereas the second kind aims at detecting good cost to step size improvements.

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.

Acknowledgements. Jacques Desrosiers acknowledges the National Science and Engineering Research Council of Canada for its financial support.

References

Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Upper Saddle River, NJ, USA, 1993.

Hatem M. T. Ben Amor, Jacques Desrosiers, and José M. Valério de Carvalho. Dual-Optimal Inequalities for Stabilized Column Generation. *Operations Research*, 54(3):454–463, 2006. doi:10.1287/opre.1060.0278.

Hatem M. T. Ben Amor, Jacques Desrosiers, and Antonio Frangioni. On the choice of explicit stabilizing terms in column generation. *Discrete Applied Mathematics*, 157(6):1167–1184, 2009. doi:10.1016/j.dam.2008.06.021.

Robert E. Bixby. Solving real-world linear programs: A decade and more of progress. *Operations Research*, 50(1): 3–15, 2002. doi:10.1287/opre.50.1.3.17780.

George B. Dantzig and Mukund N. Thapa. *Linear Programming 1: Introduction*. Springer Series in Operations Research and Financial Engineering. Springer, New York, NY, USA, 1997.

George B. Dantzig and Mukund N. Thapa. *Linear Programming 2: Theory and Extensions*. Springer Series in Operations Research and Financial Engineering (Book 2). Springer, New York, NY, USA, 2003.

George B. Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960. doi:10.1287/opre.8.1.101.

Olivier du Merle, Daniel Villeneuve, Jacques Desrosiers, and Pierre Hansen. Stabilized column generation. *Discrete Mathematics*, 194:229–237, 1999. doi:10.1016/S0012-365X(98)00213-1.

Issmail Elhallaoui, Daniel Villeneuve, François Soumis, and Guy Desaulniers. Dynamic aggregation of set partitioning constraints in column generation. *Operations Research*, 53(4):632–645, 2005. doi:10.1287/opre.1050.0222.

Issmail Elhallaoui, Guy Desaulniers, Abdelmoutalib Metrane, and François Soumis. Bi-dynamic constraint aggregation and subproblem reduction. *Computers & Operations Research*, 35(5):1713–1724, 2008. doi:10.1016/j.cor.2006.10.007.

Issmail Elhallaoui, Abdelmoutalib Metrane, Guy Desaulniers, and François Soumis. An Improved Primal Simplex algorithm for degenerate linear programs. *INFORMS Journal on Computing*, 23:569–577, 2011. doi:10.1287/ijoc.1100.0425.

John J. Forrest and Donald Goldfarb. Steepest-edge simplex algorithms for linear programming. *Mathematical Programming*, 57(1):341–374, 1992. doi:10.1007/BF01581089.

Jean Bertrand Gauthier, Jacques Desrosiers, and Marco E. Lübbecke. Decomposition theorems for linear programs. *Operations Research Letters*, 42(8):553–557, December 2014. doi:10.1016/j.orl.2014.10.001.

Jean Bertrand Gauthier, Jacques Desrosiers, and Marco E. Lübbecke. About the minimum mean cycle-canceling algorithm. *Discrete Applied Mathematics*, 196:115–134, 2015a. doi:10.1016/j.dam.2014.07.005. Advances in Combinatorial Optimization.

Jean Bertrand Gauthier, Jacques Desrosiers, and Marco E. Lübbecke. Tools for primal degenerate linear programs: IPS, DCA, and PE. *EURO Journal on Transportation and Logistics*, pages 1–44, 2015b. doi:10.1007/s13676-015-0077-5.

Jean Bertrand Gauthier, Jacques Desrosiers, and Marco E. Lübbecke. A strongly polynomial Contraction-Expansion algorithm for network flow problems. Les Cahiers du GERAD G-2016-18, HEC Montréal, Montreal, QC, Canada, March 2016.

Andrew V. Goldberg and Robert Endre Tarjan. Finding minimum-cost circulations by canceling negative cycles. *Journal of the ACM*, 36(4):873–886, 1989. doi:10.1145/76359.76368.

Timo Gschwind and Stefan Irnich. Dual inequalities for stabilized column generation revisited. *INFORMS Journal on Computing*, 28(1):175–194, 2016. doi:10.1287/ijoc.2015.0670.

Paula M. J. Harris. Pivot selection methods of the Devex LP code. *Mathematical Programming*, 5(1):1–28, 1973. doi:10.1007/BF01580108.

Marco E. Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations Research*, 53(6): 1007–1023, 2005. doi:10.1287/opre.1050.0234.

Samuel Rosat, Issmail Elhallaoui, François Soumis, and Driss Chakour. Influence of the normalization constraint on the integral simplex using decomposition. *Discrete Applied Mathematics*, 2016a. doi:10.1016/j.dam.2015.12.015.

Samuel Rosat, Frédéric Quesnel, Issmail El Hallaoui, and François Soumis. Dynamic penalization of fractional directions in the integral simplex using decomposition: Application to aircrew scheduling. Les Cahiers du GERAD G-2016-01, HEC Montréal, Montreal, QC, Canada, January 2016b.

Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., Chichester, West Sussex, England, 1986.

Mehdi Towhidi, Jacques Desrosiers, and François Soumis. The positive edge criterion within COIN-OR's CLP. *Computers & Operations Research*, 49(0):41–46, 2014. doi:10.1016/j.cor.2014.03.020.

José M. Valério de Carvalho. Using extra dual cuts to accelerate convergence in column generation. *INFORMS Journal on Computing*, 17(2):175–182, 2005. doi:10.1287/ijoc.1030.0060.

Abdelouahab Zaghrouti, François Soumis, and Issmail El Hallaoui. Integral Simplex Using Decomposition for the Set Partitioning Problem. *Operations Research*, 62(2):435–449, 2014. doi:10.1287/opre.2013.1247.