Les Cahiers du GERAD ISSN: 0711–2440

Integral simplex using decomposition with primal cutting planes

S. Rosat, I. Elhallaoui,
F. Soumis, A. Lodi

G–2015–44

May 2015

Les textes publiés dans la série des rapports de recherche *Les Cahiers du GERAD* n'engagent que la responsabilité de leurs auteurs.

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2015.

The authors are exclusively responsible for the content of their research papers published in the series *Les Cahiers du GERAD*.

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2015.

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Integral simplex using decomposition with primal cutting planes

Samuel Rosat [a]
Issmail Elhallaoui [a]
François Soumis [a]
Andrea Lodi [b]

[a] *GERAD & Polytechnique Montréal, Montréal (Québec) Canada, H3C 3A7*

[b] *DEI, Università di Bologna, 40136 Bologna, Italy*

samuel.rosat@gerad.ca
issmail.elhallaoui@gerad.ca
francois.soumis@gerad.ca
andrea.lodi@unibo.it

May 2015

Les Cahiers du GERAD
G–2015–44

Copyright © 2015 GERAD

Abstract: We propose a primal algorithm for the Set Partitioning Problem based on the Integral Simplex Using Decomposition of Zaghrouti et al. (2014). We present the algorithm in a pure primal form, and relate it to other augmenting methods. We show that cutting planes can be transferred to the complementary problem, and we characterize the set of transferable cuts as a nonempty subset of primal cuts that are tight to the current solution. We prove that these cutting planes always exist, we propose efficient separation procedures for primal clique and odd-cycle cuts, and prove that their search space can be restricted to a small subset of the variables making the computation efficient. Numerical results demonstrate the effectiveness of adding cutting planes to the algorithm; tests are performed on small and large-scale set partitioning problems from aircrew and bus-driver scheduling instances up to 1,600 constraints and 570,000 variables.

Key Words: 0/1 programming, integral simplex, primal algorithms, set partitioning, primal cutting-planes, scheduling.

Acknowledgments: This work was supported by a Collaborative Research and Development grant from the Natural Sciences and Engineering Research Council of Canada (NSERC) and Kronos Inc. Samuel Rosat benefitted of a grant from the International Internship Program of the Fonds de Recherche du Québec Nature et Technologies (FRQNT).

1 Introduction

Introduced in 1969 by Garfinkel et al. [9], the Set Partitioning Problem (SPP) is a well known model of integer linear programming. Its popularity mainly comes from its simple expression, and the wide range of its applications. It can be expressed as[1]

SPP	Given a set $\mathcal{X}$ and a set of its subsets, $\mathcal{X}_1, \ldots, \mathcal{X}_n \subseteq \mathcal{X}$ of respective cost $c_1, \ldots, c_n$, determine a partition of $\mathcal{X}$, using only some of the $\mathcal{X}_i$, $1 \leq i \leq n$, and of minimum (or maximum) cost.

Applications range from aircrew scheduling [5] to vehicle routing [2] and electricity production planning [28], among others.

The most common method for integer linear programming, and thus SPP, is the *branch and bound*, originally introduced in [4]. That method is based on the recursive exploration of a branching tree to determine an optimal solution. Even though pruning some branches may be possible by computing lower/upper bounds with the linear relaxation, and even though standard cutting planes can tighten this linear relaxation and speed up the process (*branch and cut*), the size of the branching tree is exponential in the number of variables and the algorithm can get very slow on large instances. Moreover, in many practical cases, the branch and bound is not even able to take efficiently advantage of available initial solutions. These observations motivate the search for other methods than branch and bound to solve integer linear programs. We are obviously neither the firsts, nor probably the lasts, to raise that issue. Alternative methods based on linear programming do indeed exist, and among them, *primal algorithms*. These algorithms, sometimes described as *augmenting methods* or *all-integer* algorithms, are based on the following pattern: given a starting feasible solution, improve it iteratively to obtain a sequence of improving feasible solutions until optimality is reached. Two of their key features are to avoid the exploration of the aforementioned branching tree and take advantage of existing starting solutions. Moreover, classical methods from integer programming can be adapted to the primal framework, provided some theoretical adjustments are taken; in this work, we particularly focus on the case of cutting planes.

In the case of {0,1}-programming, all integer points of the linear relaxation of the {0,1}-program are extreme points of its linear relaxation. Therefore, one can design an all-integer algorithm such that all improvements are obtained by performing simplex pivots. Such algorithms are hence named *integral simplex* algorithms. One of the main drawbacks of algorithms based on simplex pivots is their inability to perform well on degenerate problems. In mathematical programming, degeneracy occurs when some basic variables are at one of their bounds, which is common in the particular case of SPP. In this case, it is very much likely that the value of variable entering the simplex basis cannot be modified without making the current solution infeasible. The resulting degenerate pivot leads to no change in the solution, and no improvement in the objective value. Recently, Zaghrouti et al. [40] proposed a new algorithm for SPP, the *Integral Simplex Using Decomposition* (ISUD) which is an offspring of recent works conducted around the *Improved Primal Simplex* in [19, 7, 36, 20]. It is therefore designed to take advantage of degeneracy, rather than suffer from it. Combined with (i) the canonically degenerate nature of the SPP, particularly in the industrial applications, and (ii) the observation that primal algorithms experience troubles with degeneracy, particularly when primal cutting planes are used (e.g., [18]), the previous observations on the advantageous way that ISUD copes with degeneracy show its potential to embody the next generation of primal algorithms. Furthermore, it seems natural to apply primal cutting planes techniques within an "anti-degeneracy" framework since degeneracy is reported as the main trouble experienced when adding cuts in primal methods.

From the applicative point of view, and as shown in the last part of this paper, ISUD proves highly efficient for *reoptimization*. This key feature makes it a very promising method in two particularly important cases which are reoptimization of existing solutions, and all-integer column generation. First, the need to quickly reoptimize an existing solution (a schedule) in case of unforeseen events is fundamental in many practical cases. Take for instance the example of airline crew scheduling (see [34]): The manpower planned schedule must often be modified to react to day-to-day operational constraints such as schedule disruptions, aircraft

[1] A formulation of SPP as a mathematical program is given in Section 2.3.

substitutions, crew absences, strikes or even volcanic eruptions! Second, consider the all-integer column generation process. Column generation is an optimization technique used to solve very large problems. When solving problems with integer variables, it is usually embedded in a branch-and-price framework. As for standard integer programming, the exploration of the branching tree can turn to be a very long process. In the same way that primal algorithms are an alternative to branch and bound, all-integer column generation is an alternative to branch and price. In all-integer column generation, a subset of the columns of the constraint matrix is generated in the beginning, and the optimal solution to this program, called *restricted master problem*, is found. Then, subproblems generate new columns to be appended to the matrix, and the new optimal solution for the extended matrix must be determined. One then iterates the process until no column can be generated, that improves the solution. Obviously, solving the extended problem from scratch (*branch and bound*) results in a loss of information and a probable lack of efficiency, while using the previous solution as a warm-start could give the algorithm a substantial advantage (*primal algorithms*). Hence, any efficient all-integer column generation code requires a good reoptimization method, since the global process is based on successive updates of an integral solution. Thus, Isud is an excellent candidate for the reoptimization of the optimal solution of the restricted master problem every time it is extended.

This paper is organized as follows. A literature review on primal algorithms, primal cutting planes and integral simplex methods, as well as some notation, problem definitions and contribution statement are given in Section 2. The Isud algorithm is presented in an innovative way, and new theoretical results are discussed in Section 3. Primal cutting planes are discussed in Section 4, new separation procedures are described, and we show that the search space of the separation can be restricted without preventing from finding these cutting planes. Numerical results are displayed in Section 5, which show the potential of adding primal cuts to Isud, and conclusions are drawn in Section 6. An extended abstract of this work was published in [27].

2 Literature review and contribution statement

2.1 Notations

Before addressing the SPP, a general introduction on primal algorithms for ILP is given here. We consider a generic integer linear program

$$z^\star_{ILP} = \min_{\boldsymbol{x}\in\mathbb{R}^n} \left\{ \boldsymbol{c}^T\boldsymbol{x} \,|\, \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\,,\, \boldsymbol{x} \geq 0 \text{ and } \boldsymbol{x} \text{ is integer} \right\} \tag{ILP}$$

where $\boldsymbol{A} \in \mathbb{N}^{m\times n}$, $\boldsymbol{b} \in \mathbb{N}^m$ and $\boldsymbol{c} \in \mathbb{N}^n$, with $\mathbb{N}$ the set of natural integers. The set of indices of the rows is denoted as $\mathcal{R} = \{1,\ldots,m\}$. $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ are called the linear constraints and $\boldsymbol{x} \geq 0$ the nonnegativity constraints. The set of all feasible solutions of ILP is denoted by $\mathcal{F}_{\text{ILP}}$. $z^\star_{ILP}$ is called the optimal value of ILP and any feasible solution $\boldsymbol{x}^\star \in \mathcal{F}_{\text{ILP}}$ such that $\boldsymbol{c}^T\boldsymbol{x}^\star = z^\star_{ILP}$ is called an optimal solution of ILP. The linear relaxation of ILP, denoted as ILP^{LR} is the linear program obtained by relaxing the integrality constraints of ILP. Its feasible domain and optimal value are resp. denoted as $\mathcal{F}_{\text{ILP}^{LR}}$ and $z^\star_{ILP^{LR}}$. Note that in this paper, all optimization models are written in their minimization form, but the ideas and formulas also apply to maximization scenarios.

Lower-case bold symbols are used for column vectors and upper-case bold symbols denote matrices. For subsets $\mathcal{X} \subseteq \{1,\ldots,m\}$ of row indices and $\mathcal{Y} \subseteq \{1,\ldots,n\}$ of column indices, the submatrix of $\boldsymbol{A}$ with rows indexed by $\mathcal{X}$ and columns indexed by $\mathcal{Y}$ is denoted as $\boldsymbol{A}_{\mathcal{X}\mathcal{Y}}$. Similarly, $\boldsymbol{A}_{\mathcal{X}\cdot}$ is the set of rows of $\boldsymbol{A}$ indexed by $\mathcal{X}$, while $\boldsymbol{A}_{\cdot\mathcal{Y}}$ is the set of columns of $\boldsymbol{A}$ indexed by $\mathcal{Y}$, and for any vector $\boldsymbol{v} \in \mathbb{R}^n$, $\boldsymbol{v}_{\mathcal{Y}}$ is the subvector of all v_y, $y \in \mathcal{Y}$. The vector of all zeros (resp. ones) with dimension dictated by the context is denoted by $\boldsymbol{0}$ (resp. $\boldsymbol{e}$), and $\boldsymbol{A}^T$ is the transpose of $\boldsymbol{A}$. Finally, the linear span of all columns of any matrix $\boldsymbol{M}$, also called image of $\boldsymbol{M}$, is denoted as $Span\,(M)$.

2.2 Primal algorithms

As noted by Letchford and Lodi [16], algorithms for integer linear programming can be divided into three classes: *dual fractional*, *dual integral*, and *primal* methods. *Dual fractional* algorithms maintain optimality

and linear-constraint feasibility at every iteration, and they stop when integrality is reached. They are typically standard cutting plane procedures such as Gomory's algorithm [12]. The classical branch-and-bound scheme is also based on a dual-fractional approach, in particular for the determination of lower bounds. *Dual integral* methods maintain integrality and optimality, and they terminate once the (primal) linear constraints are satisfied. Letchford and Lodi give the sole example of another algorithm of Gomory [13]. Finally, *primal algorithms* maintain feasibility (including integrality) throughout the process and stop when optimality is reached. These are in fact descent algorithms for which the improving sequence $(\boldsymbol{x}_k)_{k=1\ldots K}$ satisfies the conditions

- C1 $\boldsymbol{x}^k \in \mathcal{F}_{\text{ILP}}$;
- C2 $\boldsymbol{x}^K$ is optimal;
- C3 $\boldsymbol{c}^T\boldsymbol{x}^{k+1} < \boldsymbol{c}^T\boldsymbol{x}^k$.

Primal methods – sometimes classified as *augmenting algorithms* – were first introduced simultaneously by Ben-Israel and Charnes [3] and Young [38] and improved by Young [39] and Glover [10]. In Young's method [38, 39], at iteration k, a simplex pivot is considered: if it leads to an integer solution, it is performed; otherwise, cuts are generated and added to the problem, thereby changing the underlying structure of the constraints. Young also developed the concept of a *augmenting* (or *improving*) vector at $\boldsymbol{x}^k$, i.e., a vector $\boldsymbol{z} \in \mathbb{R}^n$ such that $\boldsymbol{x}^k + \boldsymbol{z}$ is integer, feasible, and of lower cost than $\boldsymbol{x}^k$. From this notion comes the *integral augmentation problem* (**iAUG**) that involves finding such a direction if it exists or asserting that $\boldsymbol{x}^k$ is optimal.

iAUG	Find an improving vector $\boldsymbol{z} \in \mathbb{N}^n$ such that $(\boldsymbol{x}^k + \boldsymbol{z}) \in \mathcal{F}_{\text{ILP}}$ and $\boldsymbol{c}^T\boldsymbol{z} < 0$ or assert that $\boldsymbol{x}^k$ is optimal for ILP.

Remark 1 Traditionally, papers on constraint aggregation and integral simplex algorithms deal with minimization problems, whereas authors usually present generic primal algorithms for maximization problems. We therefore draw the reader's attention to the following: to retain the usual classification, we call the improving direction problem **iAUG**, although it supplies a decreasing direction. In the same way, an improvement (general term) is, in our case, a decrease. For the same reasons, we still call it an augmentation.

For the sake of readability, in this paper, we will differentiate **iAUG** (above) from the fractional augmentation **fAUG**. The latter is the relaxation of the former in which $\boldsymbol{z}$ may be fractional and $\boldsymbol{x}^k + \boldsymbol{z}$ needs only be a solution of the linear relaxation ILP^{LR}.

In the end of the 1990s, there has been a renewed interest in primal integer algorithms, inspired by Robert Weismantel as mentioned in [17]. Many recent works specifically concern 0/1-LP (integer programming problems for which the variables can only take values 0 or 1). However, only a few papers have addressed the practical solution of **iAUG**, most of them considering it as an oracle. As a matter of fact, most of the rare computational work since 2000 on primal algorithms concerns the *primal separation problem*, defined as

P-SEP	Given a feasible solution $\boldsymbol{x}^k \in \mathcal{F}_{\text{ILP}}$ and an infeasible point $\boldsymbol{x}^\star$, find a hyperplane that separates $\boldsymbol{x}^\star$ from $\mathcal{F}_{\text{ILP}}$ and that is tight at $\boldsymbol{x}^k$ or assert that none exists.

In this case, $\boldsymbol{x}^\star$ is typically a vertex of the feasible domain of the linear relaxation $\mathcal{F}_{\text{ILP}^{LR}}$ obtained by performing one or several simplex pivots from $\boldsymbol{x}^k$. An example of primal separation hyperplanes is given in Figure 1.

In 2003, Eisenbrand et al. [6] proved that the primal separation problem is, from the theoretical point of view, as difficult as the integral optimization problem for 0/1-LP. It is therefore expected to be a "complicated" problem because 0/1-LP is $\mathcal{NP}$-hard. Letchford and Lodi [16, 18] and Eisenbrand et al. [6] adapt well-known algorithms for the standard separation problem to primal separation. To the best of our knowledge, only few papers present computational experiments using primal methods. Best examples are those of

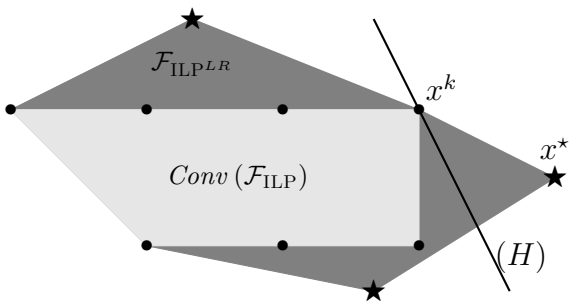


Figure 1: Example of primal separation. (H) is a primal cut separating $\boldsymbol{x}^\star$ from $Conv\,(\mathcal{F}_{\text{ILP}})$, and tight at $\boldsymbol{x}^k$. The dark area represents the feasible domain of the linear relaxation $\mathcal{F}_{\text{ILP}^{LR}}$. The feasible domain of ILP is a finite set represented by bullets ($\bullet$) and its convex hull $Conv\,(\mathcal{F}_{\text{ILP}})$ is the light grey polyhedron.

Salkin and Koncal [29], Letchford and Lodi [16], Haus et al. [14], and Stallmann and Brglez [33]. All these papers present results on small to mid-size instances. Haus et al. [14] describe a solid framework and their implementation is certainly the most complete one. Letchford and Lodi [16] present results for an algorithm using primal cutting planes and, interestingly, they stated that degeneracy prevented them from solving larger instances. As was already mentioned above, the ISUD algorithm was originally designed to cope with degeneracy and therefore seems a promising answer to these computational limits of primal algorithms.

For further information on primal algorithms, the reader is referred to the more extensive review of Spille et al. [32].

2.3 The Set Partitioning Problem

The Set Partitioning Problem (SPP) is a particular case of 0/1-LP, presented in the introduction. In a mathematical programming form, it reads as

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \left\{ \boldsymbol{c}^T\boldsymbol{x} \,|\, \boldsymbol{A}\boldsymbol{x} = \boldsymbol{e}\,, \boldsymbol{0} \leq \boldsymbol{x} \leq \boldsymbol{e} \text{ and } \boldsymbol{x} \text{ is integer} \right\} \tag{SPP}$$

where $\boldsymbol{A} \in \{0,1\}^{m\times n}$ is a $\{0,1\}$-matrix, and $\boldsymbol{c} \in \mathbb{N}^n$ is any integer cost vector. Obviously, given the bounds ($\boldsymbol{0}$ and $\boldsymbol{e}$) and the integrality constraints, $\mathcal{F}_{\text{SPP}}$ only contains $\{0,1\}$-vectors. As for any integer linear program, its linear relaxation SPP^{LR} is defined by relaxing the integrality constraints and the feasible domain of this linear relaxation is denoted as $\mathcal{F}_{\text{SPP}^{LR}}$. Moreover, given a current solution $\boldsymbol{x}^0 \in \mathcal{F}_{\text{SPP}}$, the indices of its components can be partitioned into sets $\mathcal{P} = \{j | x^0_j = 1\}$, which is the set of positive-valued variables, and $\mathcal{Z} = \{j | x^0_j = 0\}$, which is the set of null variables. Submatrix $\boldsymbol{A}_{.\mathcal{P}}$ is referred to as the *working basis*, and so is $\mathcal{P}$ by extension. Its indices and the variables of $\boldsymbol{x}_{\mathcal{P}}$ are respectively referred to as *basic indices* and *basic variables*, and $p = |\mathcal{P}|$ denotes the cardinality of this working basis. The working basis is different from a standard simplex basis because it only contains linearly independent positive-valued variables (no degenerate variables). In the rest of this paper, the terms basis, basic, etc. refer to the working basis $\mathcal{P}$.

As proved in 1969 by Trubin [37], the SPP is *quasi-integral*, i.e., every edge of the convex hull of the feasible set $Conv\,(\mathcal{F}_{\text{SPP}})$ is also an edge of the polytope of the linear relaxation $\mathcal{F}_{\text{SPP}^{LR}}$. A consequence of this property is the existence of a decreasing sequence of integer solutions of SPP leading to an optimal solution, such that two consecutive solutions are adjacent vertices of $\mathcal{F}_{\text{SPP}^{LR}}$ (easily proven by applying the simplex algorithm over $Conv\,(\mathcal{F}_{\text{SPP}})$). When working on the SPP, the following condition `C4` can therefore be added to `C1`–`C3` to transform a primal algorithm into an *integral simplex* without preventing the procedure to reach optimality.

`C4` x^{k+1} is a neighbor of x^k in $\mathcal{F}_{\text{SPP}^{LR}}$.

These methods, first introduced in 1975 by Balas and Padberg [1], yield a sequence of improving all-integer solutions, obtained by only performing simplex pivots. Since this seminal paper, other integral simplex

methods have been proposed (see Thompson [35] and Saxena [30]). Amongst contemporary work conducted parallel to ours, the most interesting one to mention is that conducted by Rönnberg and Larsonn (see [24, 25]), and Rönnberg's PhD Thesis [23] that tackles nurse scheduling problems with an integral simplex algorithm applied within a column-generation framework. Finally, note that the SPP is by nature highly degenerate. Hence it seems relevant to apply that kind of "anti-degeneracy" techniques in this case.

2.4 Contribution statement

With the concepts introduced in Section 2, we can describe the contributions of Sections 3 and 4 more clearly. In Section 3, we present the ISUD algorithm in a primal way, and relate it to the augmentation problems **iAUG** and **fAUG**. We formulate **fAUG** as a linear program, and **iAUG** as a nonlinear one of which **fAUG** is the linear relaxation. Each of these problems is decomposed into two subproblems. We reduce the number of constraints of both decomposed subproblems, and give a geometrical interpretation of these row-reduced problems. We also provide a simple characterization of integer directions. Section 4 addresses the improvement of the linear relaxation of **iAUG** with cutting planes. We demonstrate that every valid inequality for **iAUG** can be obtained as the linear transformation of a primal cut of SPP. We prove that such a primal cut always exists and that the characterization of integer directions given in Section 3 remains correct after the addition of cutting planes. Finally, we introduce two new **P-SEP** procedures for primal clique and odd-cycle cuts, and show that the search space for the cuts can be reduced to a small number of variables without changing the outcome of **P-SEP**.

3 The Integral Simplex Using Decomposition (ISUD)

This section aims to present the Integral Simplex Using Decomposition (ISUD) of Zaghrouti et al. [40] from a purely primal point of view, so as to make the link with primal algorithms straightforward, and simplify some proofs as well as the geometrical interpretation of the process. Section 3.1 concentrates on **fAUG**, while Section 3.2 also considers integrality and tackles **iAUG**.

Hereinafter, suppose a decreasing sequence of solutions of SPP ending at $\boldsymbol{x}^k$ is known, and **iAUG** must now be solved. For the sake of readability, we always denote the current (binary) solution as $\boldsymbol{x}^0$. We want to determine a direction $\boldsymbol{d} \in \mathbb{R}^n$ and a step $r > 0$ such that $\boldsymbol{x}^1 = \boldsymbol{x}^0 + r\boldsymbol{d} \in \mathcal{F}_{\text{SPP}}$ and of lower cost than $\boldsymbol{x}^0$ or to assert that $\boldsymbol{x}^0$ is optimal. The set of positive-valued variables is denoted as $\mathcal{P}^0 = \{j | x^0_j = 1\}$, and that of null variables as $\mathcal{Z}^0 = \{j | x^0_j = 0\}$. For the sake of readability, $\mathcal{P}$, $\mathcal{Z}$, $\boldsymbol{d}$, and r (and later other objects) will not be indexed on the index of the current solution (k or 0) although they depend on $\boldsymbol{x}^0$ (or $\boldsymbol{x}^k$).

3.1 Fractional augmentation fAUG and phase decomposition

In Section 3.1, the sole problem of a fractional augmentation, **fAUG**, is considered. However, for the sake of clarity, $\boldsymbol{x}^0 \in \mathcal{F}_{\text{SPP}}$ is still supposed to be integer. This section extends the work done by Omer et al. [20], and Rosat et al. [26] and details it in the case of the SPP.

3.1.1 Generic fractional augmentation

To practically address **fAUG**, it must be formulated in such a way that it can algorithmically be solved. It consists in finding a direction $\boldsymbol{d} \in \mathbb{R}^n$ such that it is *feasible* ($\exists \rho > 0 | \boldsymbol{x}^k + \rho \boldsymbol{d} \in \mathcal{F}_{\text{SPP}^{LR}}$) and *augmenting* ($\boldsymbol{c}^T \boldsymbol{d} < 0$). The set of all feasible directions at $\boldsymbol{x}^0$ is the cone

$$\Gamma = \{\boldsymbol{d} \in \mathbb{R}^n \mid \boldsymbol{A}\boldsymbol{d} = 0\,,\ \boldsymbol{d}_{\mathcal{P}} \leq \boldsymbol{0}\,,\ \boldsymbol{d}_{\mathcal{Z}} \geq \boldsymbol{0}\}\,, \tag{1}$$

from which we will only consider a section,

$$\Delta = \Gamma \cap \left\{\boldsymbol{d} \in \mathbb{R}^n \mid \boldsymbol{e}^T \boldsymbol{d}_{\mathcal{Z}} = 1\right\}. \tag{2}$$

The linear constraint $\boldsymbol{e}^T \boldsymbol{d}_{\mathcal{Z}} = 1$ is called the normalization constraint. A geometric interpretation of Γ and Δ is given in Figure 2. Note that, since $\boldsymbol{A}$ is nonnegative and because of the sign constraints, any nonzero

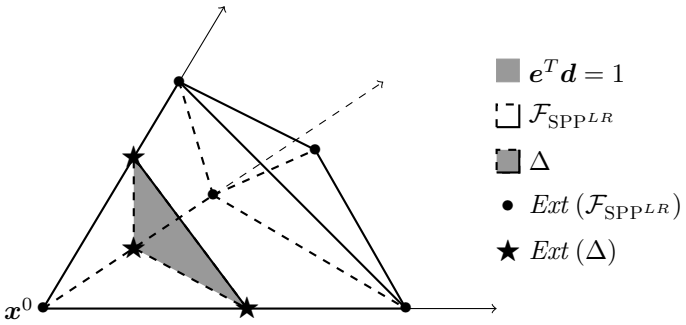


Figure 2: Geometric description of Δ and Γ. The cone Γ of all feasible directions at $\boldsymbol{x}^0$ is defined by the three arrowed lines. The grey area represents Δ, the set of normalized feasible directions.

feasible direction $\boldsymbol{d} \in \Gamma$ has nonzero terms in both $\boldsymbol{d}_{\mathcal{P}}$ and $\boldsymbol{d}_{\mathcal{Z}}$. Hence, the normalization constraint defines a proper section of the cone and Δ is a (bounded) polytope.

It is easy to see that at least one feasible direction is augmenting if and only if the program

$$z^{\star}_{\text{MIMA}} = \min_{d \in \mathbb{R}^n} \left\{ \boldsymbol{c}^T \boldsymbol{d} \,|\, \boldsymbol{d} \in \Delta \right\} \tag{MIMA}$$

satisfies $z^{\star}_{\text{MIMA}} < 0$. On the one hand, any optimal solution of MIMA yields a solution to **fAUG**; on the other hand, if $z^{\star}_{\text{MIMA}}$ is nonnegative, no feasible augmenting direction exists and $\boldsymbol{x}^0$ is optimal for SPP. Finally, since Δ is a polytope, MIMA is a bounded linear program. The name MIMA stands for *Maximum Incoming Mean Augmentation*: The normalization constraint only concerns incoming variables, so the objective is a mean over a reduced subset of the variables. The choice of this normalization constraint follows that of the original algorithm of Zaghrouti et al. [40].

Remark 2 The specific augmentation problem MIMA is close to the classical *maximum mean augmentation* problem (which is itself an extension of the *minimum mean cycle* [11] to more general linear programs; see [32]). They only differ in the normalization constraint: that of MIMA concerns only the nonbasic variables, while that of MMA is $-\boldsymbol{e}^T \boldsymbol{d}_{\mathcal{P}} + \boldsymbol{e}^T \boldsymbol{d}_{\mathcal{Z}} = 1$. The study of the influence of the choice of the normalization weights on the behavior of the algorithm is discussed in [26]. Hence, no further attention will be given to this matter here.

Once an optimal solution $\boldsymbol{d}^{\star}$ to MIMA has been found, the idea of an augmentation algorithm is to follow that direction as far as possible while remaining feasible. The *maximal feasible step* alongside direction $\boldsymbol{d}$ is thus defined as $r(\boldsymbol{d}) = \max\left\{\rho > 0 \,|\, \boldsymbol{x}^0 + \rho \boldsymbol{d} \in \mathcal{F}_{\text{SPP}^{LR}}\right\}$. From $\boldsymbol{x}^0$, fractional augmentation can therefore be performed as $\boldsymbol{x}^1 = \boldsymbol{x}^0 + r(\boldsymbol{d})\boldsymbol{d}$.

3.1.2 Incompatibility degree of the nonbasic variables

To decompose MIMA into smaller problems, both in terms of number of variables and constraints, the notion of incompatibility degree introduced by Elhallaoui et al. [8] must be presented here. Given a column $\boldsymbol{A}_{\cdot j}$ of the constraint matrix, a row $r \in \mathcal{R}$ is said to be *covered* by that column if $A_{rj} = 1$. For each column $\boldsymbol{A}_{\cdot j}$ of $\boldsymbol{A}$, let $\mathcal{R}_j = \{r \in \mathcal{R} \,|\, A_{rj} = 1\}$ be the set of rows covered by that column. By definition, the sets of the rows covered by the columns of $\mathcal{P}$, i.e., $\{\mathcal{R}_l\}_{l \in \mathcal{P}}$, form a partition of $\mathcal{R}$ (see Figure 3). Assume that a total order $\preceq$ is known over the indices of the rows $\mathcal{R}$. For any j, that order is extended to $\mathcal{R}_j$, and the elements of $\mathcal{R}_j$ are written according to $\preceq$ as

$$\mathcal{R}_j \,:\, r^j_1 \preceq r^j_2 \preceq \ldots \preceq r^j_{|\mathcal{R}_j|} \,.$$

Definition 1 (Incompatibility degree) The *incompatibility degree* of a column $\boldsymbol{A}_{\cdot j}$, $j \in \{1,\dots,n\}$, with respect to the working basis $\mathcal{P}$ is computed as

$$\iota_j^{\mathcal{P}} = \sum_{l \in \mathcal{P}} \sum_{t=1}^{|\mathcal{R}_l|-1} \kappa_{lj}^t \tag{3}$$

where $\kappa_{lj}^t = 1$ if $\boldsymbol{A}_{\cdot j}$ covers r_t^j or r_{t+1}^j but not both, 2 if $\boldsymbol{A}_{\cdot j}$ covers r_t^j and r_{t+1}^j but not consecutively, 0 otherwise. Here, r_t^j and r_{t+1}^j denote two rows covered by column l that are performed consecutively within that column. An example is given in Figure 3.

$\boldsymbol{A} =$

	$\mathcal{P}$		$\mathcal{Z}$				
	1	0	1	1	1	1	1
	1	0	0	1	1	1	1
	0	1	0	0	1	1	1
	0	1	0	0	0	1	1
	1	0	0	0	0	0	1
$\iota_j^{\mathcal{P}}$	0	0	1	1	2	1	2

$\hat{\boldsymbol{A}} =$

	$\mathcal{P}$		$\mathcal{Z}$				
	0	1	0	0	1	1	1
	0	1	0	0	0	1	1
	1	0	0	1	1	1	1
	1	0	0	0	0	0	1
	1	0	1	1	1	1	1
$\iota_j^{\mathcal{P}}$	0	0	1	2	3	2	0

Figure 3: Incompatibility degree with respect to $\mathcal{P}$ on a 5-rows example for two different ordering of the rows of the same matrix, namely $(1,2,3,4,5)$ in matrix $\boldsymbol{A}$, and $(3,4,2,5,1)$ in matrix $\hat{\boldsymbol{A}}$.

Remark 3 Any ordering of the rows can be used with these definitions. In scheduling applications, the rows correspond to tasks to carry out. It is therefore intuitive to order them by starting time and an incompatibility will correspond to the breaking of a succession of tasks that are performed consecutively in the current solution $\boldsymbol{x}^0$.

For the sake of clarity, we will always consider that the nonzero entries of a column of the current solution are consecutive within $\mathcal{R}$, i.e., given any pair of different indices $i, j \in \mathcal{P}$, either $\forall r \in \mathcal{R}_i, \forall r' \in \mathcal{R}_j, r_i \preceq r'_j$, or $\forall r \in \mathcal{R}_i, \forall r' \in \mathcal{R}_j, r'_j \preceq r_i$. Note that this is not the case of the left part of Figure 3. With $\iota_j^{\mathcal{P}}$ as defined in Formula (3), $\boldsymbol{A}_{\cdot j}$ is said to be $\iota_j^{\mathcal{P}}$*-incompatible*. 0-incompatible columns are called *compatible* and the others are called *incompatible*. These notions extend to the index of the column and to the corresponding variable.

Observation 1 *All columns from the working basis are compatible;*

Observation 2 *An incompatible column is one that breaks the partition of the rows* $\{\mathcal{R}_l\}_{l \in \mathcal{P}}$.

Given a solution $\boldsymbol{x}^0$, we define the following subsets of nonbasic variables $\mathcal{Z}$ as

$$\mathcal{C} = \left\{ j \in \mathcal{Z} \,|\, \iota_j^{\mathcal{P}} = 0 \right\} \text{ and } \mathcal{I}^{\iota} = \left\{ j \in \mathcal{Z} \,|\, 1 \leq \iota_j^{\mathcal{P}} \leq \iota \right\}, \forall 1 \leq \iota \leq k_{max} \tag{4}$$

Then, $\mathcal{C}$ is called the *compatible* set and the corresponding indices and variables are called *compatible*. Thus, $\mathcal{I}^{\iota}$ is called the *at most* ι*-incompatible* set and the corresponding indices and variables are said to be *at most* ι*-incompatible*. By definition, $\mathcal{I}^1 \subseteq \mathcal{I}^2 \subseteq \ldots \subseteq \mathcal{I}^{\iota_{max}} = \mathcal{I}$. $\mathcal{I}$ is called the incompatible set, and its elements and the corresponding variables are called incompatible. Finally, $(\mathcal{P}, \mathcal{C}, \mathcal{I})$ form a partition of $\{1,\dots,n\}$.

Lemma 1 *Given* $j \in \mathcal{Z}$*, the following statements are equivalent:*

(i) $\boldsymbol{A}_{\cdot j}$ *is compatible;*

(ii) $\exists \mathcal{P}_j \subseteq \mathcal{P} \,|\, \boldsymbol{A}_{\cdot j} = \sum_{l \in \mathcal{P}_j} \boldsymbol{A}_{\cdot l}$*;*

(iii) $\boldsymbol{A}_{\cdot j} \in$ *Span* $(\boldsymbol{A}_{\cdot \mathcal{P}})$.

Proof. First, $(i) \Leftrightarrow (ii)$ is a straightforward consequence of Observation 2. Second, $\boldsymbol{A}$ is a {0,1}-matrix, and $\boldsymbol{A}_{\cdot \mathcal{P}}$ forms a partition of its rows, therefore $(ii) \Leftrightarrow (iii)$. □

Lemma 1 does not apply to the left part of the example given in Figure 3 because the rows are not ordered properly in that case. The notion of compatible/incompatible column is extended to all vectors of $\mathbb{R}^m$, namely $\boldsymbol{w} \in \mathbb{R}^m$ is compatible if and only if $\boldsymbol{w} \in \mathit{Span}\,(\boldsymbol{A}_{\cdot\mathcal{P}})$. In the theoretical part of this paper, we will consider the partition of $\{1,\ldots,n\}$ into $(\mathcal{P},\mathcal{C},\mathcal{I})$. However it is algorithmically efficient to look first for an augmenting direction over $\mathcal{C}$, and then, successively $\mathcal{I}^1$, $\mathcal{I}^2$, …, until $\mathcal{I}$.

3.1.3 The IPS decomposition

As previously mentioned, the set of indices of the variables $\{1,\ldots,n\}$ is partitioned as $(\mathcal{P},\mathcal{C},\mathcal{I})$. With $\bar{\mathcal{P}} = \{1,\ldots,m\} \setminus \mathcal{P}$ and reordering the rows and columns of $\boldsymbol{A}$ so that $\mathcal{R}$ is partitioned into $(\mathcal{P},\bar{\mathcal{P}})$, we can write

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I}_p & \boldsymbol{A}_{\mathcal{P}\mathcal{C}} & \boldsymbol{A}_{\mathcal{P}\mathcal{I}} \\ \boldsymbol{A}_{\bar{\mathcal{P}}\mathcal{P}} & \boldsymbol{A}_{\bar{\mathcal{P}}\mathcal{C}} & \boldsymbol{A}_{\bar{\mathcal{P}}\mathcal{I}} \end{bmatrix} \tag{5}$$

where $\boldsymbol{I}_p$ is the $p \times p$ identity matrix. Given $\boldsymbol{d} \in \Delta$, from the constraints $\boldsymbol{A}\boldsymbol{d} = \boldsymbol{0}$, one can easily see that the aggregation of all columns corresponding to increasing variables (for which $d_j \geqslant 0$) $\boldsymbol{w} = \boldsymbol{A}_{\cdot\mathcal{Z}}\boldsymbol{d}_{\mathcal{Z}}$ is compatible. As in a reduced-gradient algorithm, we are in fact looking for an aggregate column $\boldsymbol{w}$ that can enter the working basis and take a positive value by lowering only some variables of $\mathcal{P}$. We introduce the following problems, respectively called *Restricted*-Mima and *Complementary*-Mima):

$$z^\star_{\text{R-Mima}} = \min_{\boldsymbol{d}\in\mathbb{R}^{p+|\mathcal{C}|}} \left\{ \boldsymbol{c}_{\mathcal{P}}^T\boldsymbol{d}_{\mathcal{P}} + \boldsymbol{c}_{\mathcal{C}}^T\boldsymbol{d}_{\mathcal{C}} \,|\, \boldsymbol{A}_{\cdot\mathcal{P}}\boldsymbol{d}_{\mathcal{P}} + \boldsymbol{A}_{\cdot\mathcal{C}}\boldsymbol{d}_{\mathcal{C}} = 0\,,\ \boldsymbol{e}_{\mathcal{C}}^T\boldsymbol{d}_{\mathcal{C}} = 1\,,\ \boldsymbol{d}_{\mathcal{C}} \geq 0 \right\} \tag{R-Mima}$$

$$z^\star_{\text{C-Mima}} = \min_{\boldsymbol{d}\in\mathbb{R}^{p+|\mathcal{I}|}} \left\{ \boldsymbol{c}_{\mathcal{P}}^T\boldsymbol{d}_{\mathcal{P}} + \boldsymbol{c}_{\mathcal{I}}^T\boldsymbol{d}_{\mathcal{I}} \,|\, \boldsymbol{A}_{\cdot\mathcal{P}}\boldsymbol{d}_{\mathcal{P}} + \boldsymbol{A}_{\cdot\mathcal{I}}\boldsymbol{d}_{\mathcal{I}} = 0\,,\ \boldsymbol{e}_{\mathcal{I}}^T\boldsymbol{d}_{\mathcal{I}} = 1\,,\ \boldsymbol{d}_{\mathcal{I}} \geq 0 \right\} \tag{C-Mima}$$

Theorem 1 $z^\star_{\text{Mima}} = \min\{z^\star_{\text{R-Mima}}, z^\star_{\text{C-Mima}}\}$.

Proof. Let $\boldsymbol{d} = (\boldsymbol{d}_{\mathcal{P}}, \boldsymbol{d}_{\mathcal{C}}, \boldsymbol{d}_{\mathcal{I}})$ be an optimal solution of Mima. If $\boldsymbol{d}_{\mathcal{C}} = \boldsymbol{0}$ or $\boldsymbol{d}_{\mathcal{I}} = \boldsymbol{0}$, $\boldsymbol{d}$ is respectively a solution of C-Mima or R-Mima and the result clearly holds.

Suppose now that $\boldsymbol{d}_{\mathcal{C}} \neq \boldsymbol{0}$ and $\boldsymbol{d}_{\mathcal{I}} \neq \boldsymbol{0}$. We first prove that $\boldsymbol{d}$ can be written as a convex combination of $\boldsymbol{u}'$, a solution of R-Mima, and $\boldsymbol{v}'$, a solution of C-Mima. By Lemma 1-(ii), the surrogate column $\boldsymbol{A}_{\cdot C}\boldsymbol{d}_{\mathcal{C}}$ can be written as a linear combination of columns of $\mathcal{P}$, $\boldsymbol{A}_{\cdot\mathcal{C}}\boldsymbol{d}_{\mathcal{C}} = -\boldsymbol{A}_{\cdot\mathcal{P}}\boldsymbol{u}'_{\mathcal{C}}$, $\boldsymbol{u}'_{\mathcal{P}} \leq \boldsymbol{0}$. Let $\boldsymbol{u} = (\boldsymbol{u}'_{\mathcal{P}}, \boldsymbol{d}_{\mathcal{C}}, \boldsymbol{0})$, and $\boldsymbol{v} = \boldsymbol{d} - \boldsymbol{u}$. Let $\alpha_{\boldsymbol{u}} = \|\boldsymbol{d}_{\mathcal{C}}\|_1 = \sum_{j\in\mathcal{C}} d_j$ and $\alpha_{\boldsymbol{v}} = \|\boldsymbol{d}_{\mathcal{I}}\|_1 = \sum_{j\in\mathcal{I}} d_j$. Moreover, let $\boldsymbol{u}' = \boldsymbol{u}/\alpha_{\boldsymbol{u}}$ and $\boldsymbol{v}' = \boldsymbol{v}/\alpha_{\boldsymbol{v}}$ be the corresponding normalized directions. Thus, $0 < \alpha_{\boldsymbol{u}}, \alpha_{\boldsymbol{v}}$ and, since $\boldsymbol{d}$ is a solution of Mima, the normalization constraint yields $\alpha_{\boldsymbol{u}} + \alpha_{\boldsymbol{v}} = 1$ because $\boldsymbol{d} \in \mathcal{F}_{\text{Mima}}$. Therefore, $\boldsymbol{d} = \alpha_{\boldsymbol{u}}\boldsymbol{u}' + \alpha_{\boldsymbol{v}}\boldsymbol{v}'$ is a convex combination of $\boldsymbol{u}'$ a solution of R-Mima and $\boldsymbol{v}'$ a solution of C-Mima.

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

Theorem 1 extends the results of Elhallaoui et al. [7] and Zaghrouti et al. [40], and it also justifies their procedures in a trivial way. This purely primal interpretation of their less-intuitive dual approach allows us to state a precise decomposition of Mima into R-Mima and C-Mima. As a consequence of Theorem 1, we will consider the pair of problems R-Mima and C-Mima instead of the more complicated Mima, and we will solve them sequentially. In particular, C-Mima will not be solved if $z^\star_{\text{R-Mima}} < 0$ because an improving direction is already known. In the next sections, we discuss how to reduce the number of rows in R-Mima and C-Mima to ease their solution. Recall that, for the moment, $\boldsymbol{x}^1 = \boldsymbol{x}^0 + r(\boldsymbol{d})\boldsymbol{d}$ may be fractional.

3.1.4 Row-reduction of R-Mima

The practical solution of the restriction of **fAUG** to the compatible variables only is based on the following proposition. Recall that for all $i \in \mathcal{C}$, $\mathcal{P}_i \subseteq \mathcal{P}$ is defined as in Lemma 1-(ii), i.e., it is the unique subset of $\mathcal{P}$ such that $\boldsymbol{A}_{\cdot i} = \sum_{j \in \mathcal{P}_i} \boldsymbol{A}_{\cdot j}$.

Proposition 1 *Given $j \in \mathcal{C}$, the* minimal direction *associated with j is the vector $\boldsymbol{\delta}^j \in \mathbb{R}^{p+|\mathcal{C}|}$ defined as*

$$\forall i \in \mathcal{P} \cup \mathcal{C}\,,\ \delta^j_i = \begin{cases} -1 & \text{if } i \in \mathcal{P}_j, \\ 1 & \text{if } i = j, \\ 0 & \text{otherwise,} \end{cases} \quad . \tag{6}$$

For any $j \in \mathcal{C}$, $\boldsymbol{\delta}^j$ is feasible and the set of all extreme points of $\mathcal{F}_{\text{R-Mima}}$ is exactly the set of the minimal directions associated with variables of $\mathcal{C}$, i.e., $\{\boldsymbol{\delta}^j | j \in \mathcal{C}\}$. Moreover, the maximal feasible step alongside direction $\boldsymbol{\delta}^j$ is $r(\boldsymbol{\delta}^j) = 1$.

Proof. First, let $j \in \mathcal{C}$. By definition of $\boldsymbol{\delta}^j$ and $\mathcal{F}_{\text{R-Mima}}$, $\boldsymbol{\delta}^j$ is obviously feasible for R-Mima.

Second, let $\boldsymbol{d}^\star = (\boldsymbol{d}^\star_{\mathcal{P}}, \boldsymbol{d}^\star_{\mathcal{C}}) \in \mathcal{F}_{\text{R-Mima}}$ be a feasible solution of R-Mima. We will show that $\boldsymbol{d}$ is a convex combination of one or several minimal solutions. Let $\boldsymbol{u} = \sum_{j \in \mathcal{C}} d^\star_j \boldsymbol{\delta}^j$. By construction, $\boldsymbol{u}_{\mathcal{C}} = \boldsymbol{d}^\star_{\mathcal{C}}$. Because all $\boldsymbol{\delta}^j$ are in $\mathcal{F}_{\text{R-Mima}}$, then the linearity constraints apply to their combination $\boldsymbol{u}$. Hence, $\boldsymbol{A}_{\cdot\mathcal{P}}\boldsymbol{u}_{\mathcal{P}} = -\boldsymbol{A}_{\cdot\mathcal{C}}\boldsymbol{u}_{\mathcal{C}} = -\boldsymbol{A}_{\cdot\mathcal{C}}\boldsymbol{d}^\star_{\mathcal{C}} = \boldsymbol{A}_{\cdot\mathcal{P}}\boldsymbol{d}^\star_{\mathcal{P}}$. The columns of the working basis $\boldsymbol{A}_{\cdot\mathcal{P}}$ are linearly independent, therefore, $\boldsymbol{u}_{\mathcal{P}} = \boldsymbol{d}^\star_{\mathcal{P}}$ and $\boldsymbol{d}^\star = \boldsymbol{u}$.

Moreover, $\boldsymbol{d} \in \mathcal{F}_{\text{R-Mima}}$ satisfies the normalization constraint $\boldsymbol{e}^T\boldsymbol{d}^\star_{\mathcal{C}} = \sum_{i \in \mathcal{C}} d^\star_i = 1$. Thus, $\boldsymbol{d}^\star = \boldsymbol{u} = \sum_{j \in \mathcal{C}} d^\star_j \boldsymbol{\delta}^j$ is a convex combination of minimal directions. Finally, given the current solution $\boldsymbol{x}^0$, for any $j \in \mathcal{C}$ and $\rho > 0$,

$$\forall i \in \mathcal{P} \cup \mathcal{C}\,,\ \left[\boldsymbol{x}^0 + \rho\boldsymbol{\delta}^j\right]_i = \begin{cases} 1-\rho & \text{if } i \in \mathcal{P}_j, \\ \rho & \text{if } i = j, \\ x^0_i & \text{otherwise.} \end{cases}$$

Therefore, considering the bounds $\boldsymbol{0} \leq (\boldsymbol{x}^0 + \rho\boldsymbol{d}) \leq \boldsymbol{e}$, the maximum possible value for ρ is $r(\boldsymbol{d}^0) = 1$. □

Corollary 1 *There exists $j \in \mathcal{P}$ such that $\boldsymbol{\delta}^j$ is an optimal solution of* R-Mima.

As a consequence of Proposition 1 and Corollary 1, solving R-Mima and following the optimal direction until a bound is reached is equivalent to pivoting the corresponding compatible variable into the working basis. These pivots are guaranteed to be nondegenerate, as proven in the following proposition.

Proposition 2 *Compatible variables are exactly those that yield nondegenerate pivots when inserted in the working basis.*

Proof. For SPP, when inserted in the working basis, a nonbasic variable yields a nondegenerate pivot iff it can be written as a nonnegative linear combination of the variables in $\mathcal{P}$ ($\boldsymbol{x}^0_{\mathcal{P}} = \boldsymbol{0}$ and $\boldsymbol{0} \leq \boldsymbol{x} \leq \boldsymbol{1}$). By Lemma 1-($iii$), compatible columns are exactly those that can be written as a linear combination of the columns of $\boldsymbol{A}_{\cdot\mathcal{P}}$. Therefore, if a column yields a nondegenerate pivot, it must be compatible. As proven in Proposition 1 pivoting a compatible variable x_j in the working basis can be interpreted as following direction $\boldsymbol{\delta}^j$. Since the corresponding maximal step is positive ($r(\boldsymbol{\delta}^j) = 1$), the pivot is nondegenerate and compatible variables all yield nondegenerate pivots. □

Proposition 3 R-Mima *is equivalent to the minimization program*

$$z^\star_{\text{R-Mima}} = z^\star_{\text{RP}} = \min_{j \in \mathcal{C}} \{\bar{c}_j\}\,, \tag{RP}$$

called reduced problem, *where $\bar{c} = \boldsymbol{c} - \boldsymbol{c}^T_{\mathcal{P}}\boldsymbol{A}_{\mathcal{P}}$. is the reduced costs vector. Moreover, given an optimal solution $j \in \mathcal{C}$, the corresponding direction is $\boldsymbol{\delta}^j$.*

Proof. Given an index $j \in \mathcal{C}$, the cost of the corresponding minimal direction in R-Mima is $\boldsymbol{c}^T\boldsymbol{\delta}^j = \boldsymbol{c}_{\mathcal{P}}^T\boldsymbol{\delta}_{\mathcal{P}}^j + \boldsymbol{c}_{\mathcal{C}}^T\boldsymbol{\delta}_{\mathcal{C}}^j$. With Equation (5), the linear constraints of R-Mima become

$$\begin{cases} \boldsymbol{I}_p\boldsymbol{\delta}_{\mathcal{P}}^j & +\boldsymbol{A}_{\mathcal{P}\mathcal{C}}\boldsymbol{\delta}_{\mathcal{C}}^j & = 0 \\ \boldsymbol{A}_{\bar{\mathcal{P}}\mathcal{P}}\boldsymbol{\delta}_{\mathcal{P}}^j & +\boldsymbol{A}_{\bar{\mathcal{P}}\mathcal{C}}\boldsymbol{\delta}_{\mathcal{C}}^j & = 0 \end{cases},$$

and, with Equation (6), the first row gives $\boldsymbol{\delta}_{\mathcal{P}}^j = -\boldsymbol{A}_{\mathcal{P}j}$, and thus, $\boldsymbol{c}^T\boldsymbol{\delta}^j = c_j - \boldsymbol{c}_{\mathcal{P}}^T\boldsymbol{A}_{\mathcal{P}j}$. Because the extreme points of $\mathcal{F}_{\text{R-Mima}}$ are the $\boldsymbol{\delta}^j$ for $j \in \mathcal{C}$, the result holds. □

The term of row-reduction refers to the following two facts. First, the linear program becomes a simple reduced-cost determination. Second, the computation of the whole improving direction is made without any matrix multiplication since $\boldsymbol{\delta}_{\mathcal{P}}^j = -\boldsymbol{A}_{\mathcal{P}j}$. This is in the spirit of a standard simplex pivot in which a reduced-cost analysis is performed, and then a system must be solved to determine the future solution $\boldsymbol{x}^{k+1}$. As in Proposition 3, in practice, the determination of $j \in \mathcal{C}$ is made as a reduced-cost analysis and exactly reproduces what a simplex pivot would be. Namely, the nonbasic variable of lowest reduced-cost $z^\star_{\text{R-Mima}}$ is determined, and then, if it satisfies $z^\star_{\text{R-Mima}} < 0$, a pivot is performed by inserting that variable into the working basis. However, if $z^\star_{\text{R-Mima}} \geq 0$, it becomes necessary to consider C-Mima to determine an augmenting direction. This process is the topic of the following section.

3.1.5 Row-reduction of C-Mima

In this section, we suppose that the compatible variables yield no improvement, or equivalently that $\mathcal{C} = \emptyset$. The first p constraints of C-Mima read $\boldsymbol{d}_{\mathcal{P}} = -\boldsymbol{A}_{\mathcal{P}\mathcal{I}}\boldsymbol{d}_{\mathcal{I}}$. As in a reduced gradient algorithm (e.g., [15]), the modification of the basic variables is inferred via a linear transformation of the increasing nonbasic variables $\boldsymbol{d}_{\mathcal{Z}}$, or in the case of C-Mima, $\boldsymbol{d}_{\mathcal{I}}$. Hence, we reduce C-Mima to an equivalent problem over the nonbasic variables of $\mathcal{I}$ only. For the sake of clarity, denote as $\Delta_{\mathcal{I}}$ the feasible domain of C-Mima, i.e., all elements of Δ that satisfy $\boldsymbol{d}_{\mathcal{C}} = 0$, and define $q = |\mathcal{I}|$. That domain can be defined by less linear constraints and variables than Δ as shown by Proposition 4.

Proposition 4 *With* $\bar{\Delta}_{\mathcal{I}} = \left\{\boldsymbol{d}_{\mathcal{I}} \in \mathbb{R}^q \,|\, \bar{\boldsymbol{A}}_{\bar{\mathcal{P}}\mathcal{I}}\boldsymbol{d}_{\mathcal{I}} = \boldsymbol{0}\,,\, \boldsymbol{e}^T\boldsymbol{d}_{\mathcal{I}} = 1\,,\, \boldsymbol{d}_{\mathcal{I}} \geq \boldsymbol{0}\right\}$, *then*

$$\Delta_{\mathcal{I}} = \left\{\boldsymbol{d} = \boldsymbol{T}\boldsymbol{d}_{\mathcal{I}} \,|\, \boldsymbol{d}_{\mathcal{I}} \in \bar{\Delta}_{\mathcal{I}}\right\} \tag{7}$$

where $\boldsymbol{T} = \begin{bmatrix} -\boldsymbol{A}_{\mathcal{P}\mathcal{I}} \\ \boldsymbol{I}_q \end{bmatrix}$ *and* $\bar{\boldsymbol{A}}_{\bar{\mathcal{P}}\mathcal{I}} = -\boldsymbol{A}_{\bar{\mathcal{P}}\mathcal{P}}\boldsymbol{A}_{\mathcal{P}\mathcal{I}} + \boldsymbol{A}_{\bar{\mathcal{P}}\mathcal{I}} = \boldsymbol{A}_{\bar{\mathcal{P}}\cdot}\boldsymbol{T}$.

Proof. Let $\boldsymbol{d} \in \mathbb{R}^n$ such that $\boldsymbol{d}_{\mathcal{C}} = \boldsymbol{0}$. Then,

$$\begin{aligned}
\boldsymbol{d} \in \Delta_{\mathcal{I}} \quad &\Leftrightarrow \quad \boldsymbol{A}_{\cdot\mathcal{P}}\boldsymbol{d}_{\mathcal{P}} + \boldsymbol{A}_{\cdot\mathcal{I}}\boldsymbol{d}_{\mathcal{I}} = 0\,,\, \boldsymbol{e}^T\boldsymbol{d}_{\mathcal{I}} = 1\,,\, \boldsymbol{d}_{\mathcal{P}} \leq \boldsymbol{0}\,,\, \boldsymbol{d}_{\mathcal{I}} \geq \boldsymbol{0} \\
&\Leftrightarrow \quad \begin{cases} \boldsymbol{d}_{\mathcal{P}} + \boldsymbol{A}_{\mathcal{P}\mathcal{I}}\boldsymbol{d}_{\mathcal{I}} = \boldsymbol{0} \\ \boldsymbol{A}_{\bar{\mathcal{P}}\mathcal{P}}\boldsymbol{d}_{\mathcal{P}} + \boldsymbol{A}_{\bar{\mathcal{P}}\mathcal{I}}\boldsymbol{d}_{\mathcal{I}} = \boldsymbol{0} \\ \boldsymbol{e}^T\boldsymbol{d}_{\mathcal{I}} = 1 \\ \boldsymbol{d}_{\mathcal{P}} \leq 0 \qquad \boldsymbol{d}_{\mathcal{I}} \geq 0 \end{cases} \\
&\Leftrightarrow \quad \begin{cases} \boldsymbol{d}_{\mathcal{P}} = -\boldsymbol{A}_{\mathcal{P}\mathcal{I}}\boldsymbol{d}_{\mathcal{I}} \\ (-\boldsymbol{A}_{\bar{\mathcal{P}}\mathcal{P}}\boldsymbol{A}_{\mathcal{P}\mathcal{I}} + \boldsymbol{A}_{\bar{\mathcal{P}}\mathcal{I}})\,\boldsymbol{d}_{\mathcal{I}} = \boldsymbol{0} \\ \boldsymbol{e}^T\boldsymbol{d}_{\mathcal{I}} = 1 \\ \boldsymbol{d}_{\mathcal{I}} \geq 0 \end{cases} \\
&\Leftrightarrow \quad \boldsymbol{d} = \boldsymbol{T}\boldsymbol{d}_{\mathcal{I}} \text{ and } \boldsymbol{d}_{\mathcal{I}} \in \bar{\Delta}_{\mathcal{I}}.
\end{aligned}$$

Hence, the proposition holds. □

The cost of such a direction can be computed as $\boldsymbol{c}^T\boldsymbol{d} = \boldsymbol{c}^T\boldsymbol{T}\boldsymbol{d}_{\mathcal{I}}$. Define $\bar{\boldsymbol{c}}^T = \boldsymbol{c}^T\boldsymbol{T} = \boldsymbol{c}_{\mathcal{I}}^T - \boldsymbol{c}_{\mathcal{P}}^T\boldsymbol{A}_{\mathcal{P}\mathcal{I}}$, and C-Mima is equivalent to the following program, called the *complementary problem*:

$$z^\star_{\text{C-Mima}} = z^\star_{\text{CP}} = \min\left\{\bar{c}^T\boldsymbol{d}_{\mathcal{I}} \,|\, \boldsymbol{d}_{\mathcal{I}} \in \bar{\Delta}_{\mathcal{I}}\right\}. \tag{CP}$$

The cost vector of CP is the reduced-costs vector associated with all incompatible variables, i.e., their own cost ($\boldsymbol{c}_{\mathcal{I}}^T$) minus the marginal impact they have on the values of the variables of $\mathcal{P}$ if they enter the reduced-basis ($-\boldsymbol{c}_{\mathcal{P}}^T \boldsymbol{A}_{\mathcal{P}\mathcal{I}}$).

Remark 4 In Proposition 1, for $j \in \mathcal{C}$, we called $\boldsymbol{\delta}^j$ a *minimal direction*. This denomination can be extended to directions of $\bar{\Delta}_{\mathcal{I}}$ (while still applying to those of $\bar{\Delta}_{\mathcal{C}}$) as follows: $\boldsymbol{d} \in \bar{\Delta}_{\mathcal{I}}$ is a *minimal direction* iff there exists no other direction whose support is strictly contained in that of $\boldsymbol{d}$. In that sense, the set of minimal directions of $\bar{\Delta}$ is exactly $Ext\left(\bar{\Delta}_{\mathcal{C}}\right) \cup Ext\left(\bar{\Delta}_{\mathcal{I}}\right)$ (see [26] for more details on the geometrical structure of Δ and $\bar{\Delta}$).

3.2 Integral augmentation iAUG

The previous section provides a method to find a fractional augmentation. If $z^\star_{\text{RP}} < 0$ or $z^\star_{\text{CP}} < 0$, then the corresponding solution of negative reduced cost $\boldsymbol{d} \in \Delta$ yields a new solution $\boldsymbol{x}^1 = \boldsymbol{x}^0 + r(\boldsymbol{d})\boldsymbol{d} \in \mathcal{F}_{\text{SPP}^{LR}}$. However, nothing guarantees that $\boldsymbol{x}^1$ is integral. In this section, we characterize directions that lead to an integral solution $\boldsymbol{x}^1$. Such directions are called *integral directions* as opposed to *fractional directions*. The set of all integral directions within Δ is denoted as Δ^{int}. These names apply to $\boldsymbol{d}$, and its restrictions $\boldsymbol{d}_{\mathcal{Z}}$, $\boldsymbol{d}_{\mathcal{C}}$ or $\boldsymbol{d}_{\mathcal{I}}$ depending on the context. We also give some insights on the structure of the set of all integral directions, and a generic framework for our algorithm.

3.2.1 Over compatible variables

Note that for any $j \in \mathcal{C}$, the feasible solution $\boldsymbol{\delta}^j$ of RP is an integral direction. Indeed, $\boldsymbol{A}_{\cdot j} = \sum_{i \in \mathcal{P}_j} \boldsymbol{A}_{\cdot i}$ and following a step of length $r(\boldsymbol{\delta}^j) = 1$ in that direction is equivalent to let j enter the working basis $\mathcal{P}$ and let $\mathcal{P}_j$ leave it. Namely,

$$\left[\boldsymbol{x}^0 + \rho\boldsymbol{\delta}^j\right]_i = \begin{cases} 1 & \text{if } i \in \mathcal{P} \setminus \mathcal{P}_j \\ 1-\rho & \text{if } i \in \mathcal{P}_j \\ \rho & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

The corresponding maximal step is $\rho = r(\boldsymbol{\delta}^j) = 1$, so $\boldsymbol{x}^1 = \boldsymbol{x}^0 + \boldsymbol{\delta}^j \in \mathcal{F}_{\text{SPP}}$ and integrality is maintained.

3.2.2 Characterization of integral directions in $\Delta_{\mathcal{I}}$

Now, let us consider the problem for C-Mima (and CP). We base our analysis on previous results of Zaghrouti et al. [40] that we first recall here. For any polyhedron P, denote as $Ext\,(P)$ the set of its extreme points (or vertices).

Definition 2 A set $\mathcal{S}$ of indices of the variables in $\{1, \ldots, n\}$ is *column-disjoint* if no pair of columns of $\boldsymbol{A}_{\cdot\mathcal{S}}$ has a common nonzero entry, i.e., $\forall i, j \in \mathcal{S}, i \neq j, \forall k \in \{1, \ldots, m\}, A_{ki}A_{kj} = 0$. This definition is extended to $\boldsymbol{A}_{\cdot\mathcal{S}}$ and $\boldsymbol{x}_{\mathcal{S}}$. Since $\boldsymbol{A}$ is binary, $\mathcal{S}$ is column-disjoint iff $\boldsymbol{A}_{\cdot\mathcal{S}}$ is a set of orthogonal columns.

Proposition 5 (Propositions 6 and 7, [40]) *Given $\boldsymbol{d} \in Ext\,(\Delta)$, $\boldsymbol{d}$ is an integral direction iff the support of $\boldsymbol{d}_{\mathcal{I}}$, denoted as $S = Supp\,(\boldsymbol{d}_{\mathcal{I}})$, is column-disjoint. In this case, $r(\boldsymbol{d}) = |S|$.*

Because SPP is quasi-integral, then the edges of $Conv\,(\mathcal{F}_{\text{SPP}})$ are edges of $\mathcal{F}_{\text{SPP}^{LR}}$, and $Ext\left(\Delta^{int}_{\mathcal{I}}\right) \subseteq Ext\,(\Delta)$. A geometrical description of this is shown on Figure 4. Since every linear program has at least one optimal solution that is also an extreme point of its feasible domain, and since the simplex algorithm guarantees to find such a solution, Proposition 5 has a strong practical interest. In the next section, we will show that it still remains valid when cutting planes or branching techniques are used. Hence, it is sufficient to test whether the solution of the relaxation (CP) is column-disjoint to determine if it is integral.

3.2.3 Algorithmic framework

Algorithm 1 is based on successive resolutions of augmenting problems either to $\mathcal{C}$ or to $\mathcal{I}^\iota$ for a certain incompatibility degree ι. CP$^\iota$ describes the restriction of CP to the columns of $\mathcal{I}^\iota$. INC_MAX describes the

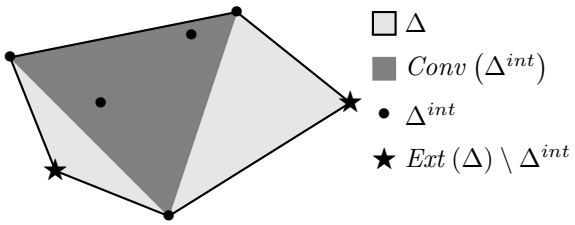


Figure 4: Geometrical insight on the extreme points of Δ and Δ^{int}. All extreme points of $Conv\left(\Delta^{int}\right)$ ($\bullet$) are extreme points of Δ. Moreover, Δ^{int} is a finite set ($\mathcal{F}_{\text{SPP}}$ is finite).

Algorithm 1: Integral Simplex Using Decomposition

Input: $\boldsymbol{x}^0$, a solution of SPP; INC_MAX, maximal incompatibility degree considered.
Output: $\boldsymbol{x}^k$, a possibly better solution of SPP.

Compute $\mathcal{P}$ and $\mathcal{C}$ associated with $\boldsymbol{x}^0$; $k \leftarrow 0$; $\iota \leftarrow 1$;
while *true* **do**
  If necessary, update $\mathcal{P}$, $\mathcal{C}$, and $\mathcal{I}^\iota$ associated with $\boldsymbol{x}^k$;
  if $z^\star_{\text{RP}} < 0$ **then**
    $\boldsymbol{\delta}^i \leftarrow$ an optimal solution of RP ($\bar{c}_i < 0$, $i \in \mathcal{C}$);
    $\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{x}^k + \boldsymbol{\delta}^i$; $k \leftarrow k+1$;
  else
    continue $\leftarrow$ true;
    while *continue = true* **do**
      If necessary, update $\mathcal{P}$, $\mathcal{C}$, and $\mathcal{I}^\iota$ associated with $\boldsymbol{x}^k$, and CP^ι;
      if $z^\star_{\text{CP}^\iota} < 0$ **then**
        $\boldsymbol{d}^\star_{\mathcal{I}} \leftarrow$ an optimal solution of CP^ι; $\boldsymbol{d}^\star \leftarrow \boldsymbol{T}\boldsymbol{d}^\star_{\mathcal{I}}$;
        if $\boldsymbol{d}^\star_{\mathcal{I}}$ *is column-disjoint* **then**
          $\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{x}^k + r(\boldsymbol{d}^\star)\boldsymbol{d}^\star$; $k \leftarrow k+1$; continue $\leftarrow$ false;
        else
          if *some stopping criterion is reached* **then**
            return $\boldsymbol{x}^k$;
          else
            - Add cutting planes to CP^ι;
            - or use branching strategy;
            - or increase ι (only if $\iota < INC\_MAX$);
            - or continue $\leftarrow$ false;
      else
        if $\iota < INC\_MAX$ **then**
          $\iota \leftarrow \iota + 1$;
        else
          return $\boldsymbol{x}^k$;

largest incompatibility degree considered during the execution. The solution returned by Algorithm 1 may not be optimal: This strongly depends on the value of INC_MAX, and on the chosen branching and cutting planes techniques. For instance, if $INC\_MAX < \iota_{max}$, there may exist augmenting integral directions involving columns of incompatibility degree greater than INC_MAX. The same holds when the branching technique is nonexhaustive.

4 Solving the augmentation problem with cutting-planes

In this section, we suppose that we know an optimal solution $\boldsymbol{d}^\star_{\mathcal{I}}$ of CP, but that this solution is not integral ($Supp\left(\boldsymbol{d}^\star_{\mathcal{I}}\right)$ is not column-disjoint). An idea to tighten the known relaxation of $\bar{\Delta}^{int}_{\mathcal{I}}$ is to add cutting planes

to CP. Given a polyhedron P, a valid inequality for P is an inequality satisfied by all elements of P; and given a point $\boldsymbol{x}'$, the inequality *separates* $\boldsymbol{x}'$ from P if it is valid for P but violated by $\boldsymbol{x}'$. Such an inequality is called a cut (or cutting plane). The main issue is to characterize valid inequalities for $\bar{\Delta}_{\mathcal{I}}^{int}$ that cut off the current optimal solution $\boldsymbol{d}_{\mathcal{I}}^{\star}$.

Notation. Denote as $\mathcal{H}_{=}^{1} = \left\{\boldsymbol{d}_{\mathcal{I}} \in \mathbb{R}^q \,|\, \boldsymbol{e}^T\boldsymbol{d}_{\mathcal{I}} = 1\right\}$ the hyperplane defined by the normalization constraint of $\bar{\Delta}_{\mathcal{I}}$. Let $\bar{\boldsymbol{\alpha}} \in \mathbb{R}^q$, $\bar{\beta} \in \mathbb{R}$, and denote by $\left(\bar{\Gamma}\right)$ the inequality

$$\left(\bar{\Gamma}\right) \,:\, \bar{\boldsymbol{\alpha}}^T\boldsymbol{d}_{\mathcal{I}} \leq \bar{\beta}. \tag{8}$$

The associated hyperplane is denoted as $\mathcal{H}_{=}^{\bar{\Gamma}} = \left\{\boldsymbol{x} \in \mathbb{R}^q | \bar{\boldsymbol{\alpha}}^T\boldsymbol{d}_{\mathcal{I}} = \bar{\beta}\right\}$ and the associated half-space as $\mathcal{H}_{\leq}^{\bar{\Gamma}} = \left\{\boldsymbol{x} \in \mathbb{R}^q | \bar{\boldsymbol{\alpha}}^T\boldsymbol{d}_{\mathcal{I}} \leq \bar{\beta}\right\}$. Without loss of generality, we can assume that $\bar{\boldsymbol{\alpha}}$ is not proportional to $\boldsymbol{e}$.

Definition 3 Let $\left(\bar{\boldsymbol{\alpha}}_1, \bar{\beta}_1\right), \left(\bar{\boldsymbol{\alpha}}_2, \bar{\beta}_2\right) \in \mathbb{R}^q \times \mathbb{R}$, and $\left(\bar{\Gamma}_1\right)$ and $\left(\bar{\Gamma}_2\right)$ be the corresponding inequalities. $\left(\bar{\Gamma}_1\right)$ and $\left(\bar{\Gamma}_2\right)$ are *equivalent* for $\bar{\Delta}_{\mathcal{I}}^{int}$ if

$$\mathcal{H}_{=}^{1} \cap \mathcal{H}_{\leq}^{\bar{\Gamma}_1} = \mathcal{H}_{=}^{1} \cap \mathcal{H}_{\leq}^{\bar{\Gamma}_2}$$

Since $\bar{\Delta}_{\mathcal{I}} \subseteq \mathcal{H}_{=}^{1}$, two equivalent inequalities exactly discard the same subset of $\bar{\Delta}_{\mathcal{I}}$ and are simultaneously valid. Hence, adding the one or the other to the formulation is equivalent.

Proposition 6 *There exists* $\bar{\boldsymbol{\alpha}}' \in \mathbb{R}^q$ *such that* $\left(\bar{\Gamma}'\right) : (\bar{\boldsymbol{\alpha}}')^T\boldsymbol{d}_{\mathcal{I}} \leq 0$ *is equivalent to* $\left(\bar{\Gamma}\right)$.

Proof. Consider $\mathcal{E}_{=} = \mathcal{H}_{=}^{1} \cap \mathcal{H}_{=}^{\bar{\Gamma}}$. The two hyperplanes $\mathcal{H}_{=}^{1}$ and $\mathcal{H}_{=}^{\bar{\Gamma}}$ are not parallel, so $\dim(\mathcal{E}_{=}) = q - 2$. $\mathbf{0} \notin \mathcal{H}_{=}^{1}$, therefore $\mathbf{0} \notin \mathcal{E}_{=}$ and $\mathcal{H}_{=}^{0} = \mathit{Span}\left(\mathcal{E}_{=} \cup \{\mathbf{0}\}\right)$ is a hyperplane containing $\mathbf{0}$. Thus, there exists $\bar{\boldsymbol{\alpha}}' \in \mathbb{R}^q$ such that $\mathcal{H}_{=}^{0}$ is defined by $\left(\bar{\boldsymbol{\alpha}}'\right)^T\boldsymbol{d}_{\mathcal{I}} = 0$. Furthermore, by construction, $\mathcal{H}_{=}^{0} \cap \mathcal{H}_{=}^{1} = \mathcal{E}_{=} = \mathcal{H}_{=}^{\bar{\Gamma}} \cap \mathcal{H}_{=}^{1}$. With $\bar{\boldsymbol{\alpha}}' = \pm\bar{\boldsymbol{\alpha}}^0$ (sign to be well chosen), the proposition holds. □

The geometrical interpretation of Proposition 6 is given on Figure 5. Hereinafter, we suppose that $\bar{\beta} = 0$. We can now characterize valid inequalities for $\bar{\Delta}_{\mathcal{I}}^{int}$.

Proposition 7 *Given* $\boldsymbol{\alpha} \in \mathbb{R}^n$ *such that* $\bar{\boldsymbol{\alpha}} = \boldsymbol{T}^T\boldsymbol{\alpha}$, $\left(\bar{\Gamma}\right)$ *is a valid inequality for* $\bar{\Delta}_{\mathcal{I}}^{int}$ *if and only if*

$$(\Gamma) \,:\, \boldsymbol{\alpha}^T\left(\boldsymbol{x} - \boldsymbol{x}^0\right) \leq 0 \tag{9}$$

is a valid inequality for $\mathcal{F}_{\text{SPP}}$.

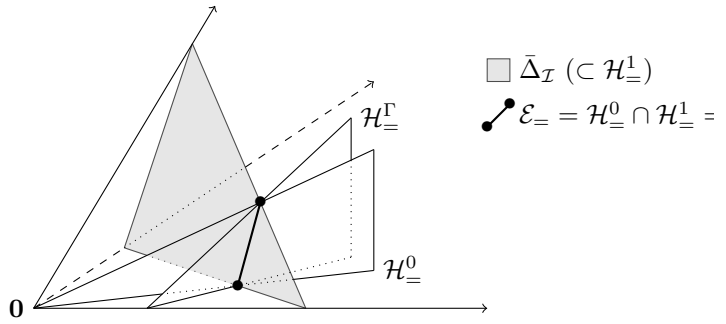


Figure 5: Equivalent inequalities. Both $\mathcal{H}_{=}^{0}$ and $\mathcal{H}_{=}^{\Gamma}$ cut off the same part of $\bar{\Delta}_{\mathcal{I}}$. $\mathcal{H}_{=}^{0}$ is of the form $\bar{\alpha}^T\boldsymbol{d}_{\mathcal{I}} \leq 0$ but not $\mathcal{H}_{=}^{\Gamma}$ ($\mathbf{0} \notin \mathcal{H}_{=}^{\Gamma}$).

Proof. Let $\boldsymbol{\alpha} \in \mathbb{R}^n$ such that $\bar{\boldsymbol{\alpha}} = \boldsymbol{T}^T\boldsymbol{\alpha}$. Recall Proposition 4: $\Delta_{\mathcal{I}} = \left\{\boldsymbol{d} = \boldsymbol{T}\boldsymbol{d}_{\mathcal{I}} \,|\, \boldsymbol{d}_{\mathcal{I}} \in \bar{\Delta}_{\mathcal{I}}\right\}$. This extends to $\Delta^{int}_{\mathcal{I}}$ and $\bar{\Delta}^{int}_{\mathcal{I}}$. Hence,

$$\begin{aligned}
\left(\bar{\Gamma}\right) \text{ is valid for } \bar{\Delta}^{int}_{\mathcal{I}} &\Leftrightarrow \forall \boldsymbol{d}_{\mathcal{I}} \in \bar{\Delta}^{int}_{\mathcal{I}}, \bar{\boldsymbol{\alpha}}^T\boldsymbol{d}_{\mathcal{I}} \leq 0 \\
&\Leftrightarrow \forall \boldsymbol{d}_{\mathcal{I}} \in \bar{\Delta}^{int}_{\mathcal{I}}, \boldsymbol{\alpha}^T\boldsymbol{T}\boldsymbol{d}_{\mathcal{I}} \leq 0 \\
&\Leftrightarrow \forall \boldsymbol{d} \in \Delta^{int}_{\mathcal{I}}, \boldsymbol{\alpha}^T\boldsymbol{d} \leq 0 \\
&\Leftrightarrow \forall \boldsymbol{d} \in \Delta^{int}_{\mathcal{I}}, \boldsymbol{\alpha}^T\left(\boldsymbol{x}^0 + r(\boldsymbol{d})\boldsymbol{d}\right) \leq \boldsymbol{\alpha}^T\boldsymbol{x}^0 \\
&\Leftrightarrow \forall \boldsymbol{x}' \in \mathcal{F}_{\text{SPP}}, \boldsymbol{\alpha}^T\left(\boldsymbol{x}' - \boldsymbol{x}^0\right) \leq 0 \\
&\Leftrightarrow (\Gamma) \text{ is valid for } SPP.
\end{aligned}$$

□

Proposition 8 *Let* (Γ) *and* $\left(\bar{\Gamma}\right)$ *be valid inequalities defined as in Proposition 7, and* $\boldsymbol{d}^\star_{\mathcal{I}} \in \Delta_{\mathcal{I}}$, $\boldsymbol{d}^\star = \boldsymbol{T}\boldsymbol{d}_{\mathcal{I}}$, *and* $\boldsymbol{x}^\star = \boldsymbol{x}^0 + r(\boldsymbol{d}^\star)\boldsymbol{d}^\star$. *Then,*

$$\left(\bar{\Gamma}\right) \textit{ separates } \boldsymbol{d}^\star_{\mathcal{I}} \textit{ from } \bar{\Delta}^{int}_{\mathcal{I}} \Leftrightarrow (\Gamma) \textit{ separates } \boldsymbol{x}^\star \textit{ from } \mathcal{F}_{\text{SPP}}$$

Proof. We only need to prove that $\left(\bar{\Gamma}\right)$ is violated by $\boldsymbol{d}^\star_{\mathcal{I}}$ if and only if (Γ) is violated by $\boldsymbol{x}^\star$. We have

$$\begin{aligned}
\bar{\boldsymbol{\alpha}}^T\boldsymbol{d}^\star > 0 &\Leftrightarrow \boldsymbol{\alpha}^T\boldsymbol{d}^\star > 0 \\
&\Leftrightarrow \boldsymbol{\alpha}^T\left(\boldsymbol{x}^0 + r(\boldsymbol{d}^\star)\boldsymbol{d}^\star\right) > \boldsymbol{\alpha}^T\boldsymbol{x}^0 \\
&\Leftrightarrow \boldsymbol{\alpha}^T\left(\boldsymbol{x}^\star - \boldsymbol{x}^0\right) > 0
\end{aligned}$$

□

What we have shown must now be seen the other way round to take advantage of previous work on primal separation. Assume that we know how to determine a primal cut (Γ) for SPP that separates $\boldsymbol{x}^\star$ from $\mathcal{F}_{\text{SPP}}$. Then, with $\bar{\boldsymbol{\alpha}} = \boldsymbol{T}^T\boldsymbol{\alpha}$, the associated inequality $\left(\bar{\Gamma}\right)$ is a cut for CP, that separates $\boldsymbol{d}^\star_{\mathcal{I}}$ from $\bar{\Delta}^{int}_{\mathcal{I}}$. Moreover, we have shown that any cut for CP can be obtained in this way. This enables us to develop a procedure based on the primal separation problem **P-SEP** – given $\boldsymbol{x}^0$ and $x^\star$, is there a valid inequality for $\mathcal{F}_{\text{SPP}}$, tight at $\boldsymbol{x}^0$, that separates $\boldsymbol{x}^\star$ from $\mathcal{F}_{\text{SPP}}$? If it exists, it will be transferred to CP to tighten the relaxation of $\bar{\Delta}^{int}_{\mathcal{I}}$. From the theoretical point of view, if $\boldsymbol{d}^\star$ is extremal, such a cut always exists as shown on Corollary 2.

Corollary 2 *Assume* $\boldsymbol{d}^\star_{\mathcal{I}} \in \textit{Ext}\left(\mathcal{F}_{\text{CP}}\right)$, *and let* $\boldsymbol{d}^\star = \boldsymbol{T}\boldsymbol{d}^\star_{\mathcal{I}}$ *(we still assume that* $\boldsymbol{d}^\star_{\mathcal{I}}$ *is not an integral direction). There always exists a valid inequality* (Γ) *tight at* $\boldsymbol{x}^0$ *that separates* $\boldsymbol{x}^\star = \boldsymbol{x}^0 + r(\boldsymbol{d}^\star)\boldsymbol{d}^\star$ *from* $\mathcal{F}_{\text{SPP}}$.

Proof. $\boldsymbol{d}^\star_{\mathcal{I}}$ is an extreme point of $\mathcal{F}_{\text{CP}}$ but does not belong to Δ^{int}. Since Δ^{int} is a finite subset of elements of $\mathcal{F}_{\text{CP}}$, and since $\mathcal{F}_{\text{CP}}$ is a polyhedron, there exists a cut $\left(\bar{\Gamma}\right)$ that separates $\boldsymbol{d}^\star_{\mathcal{I}}$ from $\bar{\Delta}^{int}_{\mathcal{I}}$. By Proposition 8, the result holds. □

It is also interesting to note that the linear transformation applied to the primal cut to transfer it to CP ($\bar{\boldsymbol{\alpha}}^T = \boldsymbol{\alpha}^T\boldsymbol{T}$) is the same as for the objective function ($\bar{\boldsymbol{c}}^T = \boldsymbol{c}^T\boldsymbol{T}$) and for the constraint matrix ($\bar{\boldsymbol{A}}_{\bar{\mathcal{P}}\mathcal{I}} = \boldsymbol{A}_{\bar{\mathcal{P}}\boldsymbol{.}}\boldsymbol{T}$). Finally, note that adding cuts does not prevent to use the characterization of extreme integer solutions as those having a column-disjoint support. Since no cutting plane is added to CP that cuts any integer directions, the set of extreme integral directions $\textit{Ext}\left(\textit{Conv}\left(\bar{\Delta}^{int}_{\mathcal{I}}\right)\right)$ is still contained in the set of extreme directions $\textit{Ext}\left(\mathcal{F}_{\text{CP}}\right)$ even after the addition of cutting planes. Geometrically, adding any valid inequality for $\textit{Conv}\left(\bar{\Delta}^{int}_{\mathcal{I}}\right)$ to the relaxation cannot transform any nonextreme integral direction into a extreme one.

4.1 Specific primal separation procedures

As exposed in the introduction, there exist previous works on the primal separation problem. It has been shown that **P-SEP** is equivalent to SPP in terms of complexity [31]. Also, there exist general-purpose families of primal cuts, such as Gomory–Young's cuts [38]. Iteratively adding cuts from these families would ultimately lead to a column-disjoint solution of CP, after a finite number of separations – provided that the choice of incoming variables follow some lexicographic order (see [38]). However, these families are not polyhedral, i.e., they do not take into account the SPP structure. We chose to study two families of inequalities that are known to yield strong cutting planes for SPP, namely Clique and Odd-cycle inequalities.

4.1.1 Primal clique inequalities

Let us consider the conflict graph of matrix $\boldsymbol{A}$, $\mathcal{G} = (\mathcal{N}, E)$, where each node of $\mathcal{N} = \{1, \ldots, n\}$ corresponds to a column of $\boldsymbol{A}$, and E is such that $\{i, j\} \in E$ iff $\boldsymbol{A}_{\cdot i}^T \boldsymbol{A}_{\cdot j} \neq 0$. Two vertices are linked by an edge if the corresponding columns are not disjoint. Given a clique $\mathcal{W}$ in this graph, any feasible solution of SPP satisfies

$$(\Gamma_{\mathcal{W}}) \; : \; \sum_{i \in \mathcal{W}} x_i \leq 1, \tag{10}$$

called the clique inequality associated with $\mathcal{W}$. Clique cuts form a family of generally strong cutting planes for SPP, and were first introduced by Padberg [22]. Given a fractional solution $\boldsymbol{x}^\star$ of SPP^{LR}, the standard clique separation problem consists in associating the weight $w_i = x_i^\star$ with each vertex of $\mathcal{G}$ and determine a clique of weight greater than 1 in $\mathcal{G}$ if any.

From the work of Letchford and Lodi [18], we will develop here a more efficient procedure for primal clique cuts. In the primal context, let $\boldsymbol{d}^\star$ be a fractional direction, and $\boldsymbol{x}^\star = \boldsymbol{x}^0 + r(\boldsymbol{d}^\star)\boldsymbol{d}^\star$. For $(\Gamma_{\mathcal{W}})$ to be tight at $\boldsymbol{x}^0$, we need $\sum_{i \in \mathcal{W}} x_i^0 = |\mathcal{W} \cap \mathcal{P}| = 1$. Hence, exactly one variable from $\mathcal{P}$ must be part of clique $\mathcal{W}$. Denote that variable as l, and the corresponding clique as $\mathcal{W}_l$. Furthermore, for $(\Gamma_{\mathcal{W}_l})$ to separate $\boldsymbol{x}^\star$ from $\mathcal{F}_{\text{SPP}}$, we must have $\sum_{i \in \mathcal{W}_l} x_i^\star > 1$. Since $x_i^\star = r(\boldsymbol{d}^\star) d_i^\star$ for all $i \in \mathcal{I}$, $\sum_{i \in \mathcal{W}_l} x_i^\star = x_l^\star + \sum_{i \in \mathcal{S}^+} x_i^\star$, where $\mathcal{S}^+ = \mathit{Supp}\,(\boldsymbol{d}_{\mathcal{I}}^\star)$. Denote also as $\mathcal{S}^- = \mathit{Supp}\,(\boldsymbol{d}_{\mathcal{P}}^\star)$ the set of columns of $\mathcal{P}$ that are impacted by direction $\boldsymbol{d}^\star$. Sets $(\mathcal{S}^-, \mathcal{S}^+)$ form a partition of $\mathit{Supp}\,(\boldsymbol{d}^\star)$ such that for all $i \in \mathit{Supp}\,(\boldsymbol{d}^\star)$, $i \in \mathcal{S}^+$ if $d_i^\star > 0$ and $i \in \mathcal{S}^-$ if $d_i^\star < 0$, so

$$\sum_{i \in \mathcal{W}_l} x_i^\star > 1 \Leftrightarrow x_l^\star + \sum_{i \in \mathcal{W}_l \cap \mathcal{S}^+} x_i^\star > 1.$$

Because $\mathcal{W}_l$ is a clique in $\mathcal{G}$, then $\mathcal{W}_l \cap \mathcal{S}^+ \subseteq \mathcal{W}_l$ is also a clique. Therefore, there exists a primal clique inequality that separates $\boldsymbol{x}^\star$ from $\mathcal{F}_{\text{SPP}}$ iff for some $l \in \mathcal{S}^-$, there exists a clique $\mathcal{W}_l$ that satisfies (1) $\mathcal{W}_l \setminus \{l\} \subseteq N_l$, (2) $l \in \mathcal{W}_l$, and (3) $w(\mathcal{W}_l) = \sum_{i \in \mathcal{W}_l} x_i^\star > 1$, where N_l is the set of neighbors of l in $\mathcal{G}$ that are also in $\mathcal{S}^+$. These properties show that it is sufficient to look for a primal clique cut within $\mathcal{S}^+ \cup \mathcal{S}^-$ to solve the complete primal clique separation problem. Hence, the primal separation procedure for primal clique cuts, called `CL_PSEP` and summarized in Algorithm 2 returns an empty set of primal clique cuts if and only if none exists.

The algorithm always finds a cut if there exists one, and that will be the most violated. However, the size of the clique could be potentially increased by adding 0-weight variables. Even if Step 4 of Algorithm 2 requires solving a $\mathcal{NP}$-hard problem,[2] note that the procedure is practically very fast because the size of N_l is usually small.

4.1.2 Primal odd-cycle inequalities

Odd-cycle inequalities form another well-known family of valid inequalities for SPP. Given a cycle $\mathcal{Q}$ of odd length in $\mathcal{G}$, the following inequality

$$(\Gamma_{\mathcal{Q}}) \; : \; \sum_{i \in \mathcal{Q}} x_i \leq \frac{|Q| - 1}{2} \tag{11}$$

[2] For the determination of the clique of maximal weight in $\mathcal{G}_l$, we use the *Cliquer* open source library, available at http://users.aalto.fi/~pat/cliquer.html, based on the algorithm described in [21].

Algorithm 2: CL_PSEP

Input: $\boldsymbol{d}^\star_{\mathcal{I}} \leftarrow$ an optimal solution of CP (fractional direction).
Output: $\mathcal{K}$, a set of primal clique cuts (empty if none exists)

$\mathcal{K} \leftarrow \emptyset$; $\boldsymbol{d}^\star \leftarrow \boldsymbol{T}\boldsymbol{d}^\star_{\mathcal{I}}$; $\mathcal{S}^- \leftarrow Supp\,(\boldsymbol{d}^\star_{\mathcal{P}})$; $\mathcal{S}^+ \leftarrow Supp\,(\boldsymbol{d}^\star_{\mathcal{I}})$;
for $l \in \mathcal{S}^-$ **do**
  $\mathcal{G}_l = (N_l, E_l) \leftarrow$ weighted subgraph of $\mathcal{G}$ induced by $N_l \subseteq \mathcal{S}^+$ $(w_i = x^\star_i)$;
  $\mathcal{W}_l \leftarrow$ clique of maximum weight in $\mathcal{G}_l$;
  $\mathcal{W}_l \leftarrow \mathcal{W}_l \cup \{l\}$;
  if $w(\mathcal{W}_l) > 1$ **then**
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\Gamma_{\mathcal{W}_l})\}$;
return $\mathcal{K}$;

is valid for SPP. Clearly, $(\Gamma_{\mathcal{Q}})$ is tight at $\boldsymbol{x}^0$ if and only if $\sum_{i\in\mathcal{Q}} x^0_i = |\mathcal{Q}\cap\mathcal{P}| = (|\mathcal{Q}|-1)/2$. Furthermore, since $\mathcal{P}$ is column-disjoint, there exists no edge between any pair of vertices of $\mathcal{P}$. Therefore, $\mathcal{Q}$ is an alternate cycle with vertices in $\mathcal{P}$ and $\mathcal{I}$, except for one $\mathcal{I}-\mathcal{I}$ edge (i.e., an edge that links two vertices of $\mathcal{I}$). Based on these observations and similarly to clique cuts, the search graph can be restricted to vertices in $Supp\,(\boldsymbol{d}^\star)$, but in a stronger manner

Proposition 9 *Every primal odd-cycle cut* $(\Gamma_{\mathcal{Q}})$ *that separates* $\boldsymbol{x}^\star$ *from* $\mathcal{F}_{\text{SPP}}$ *satisfies* $\mathcal{Q} \subseteq Supp\,(\boldsymbol{d}^\star)$.

Proof. Suppose that the result is false and that there exists a cycle $\mathcal{Q} \nsubseteq Supp\,(\boldsymbol{d}^\star)$. Let $\mathcal{S}^- = Supp\,(\boldsymbol{d}_{\mathcal{P}})$ and $\mathcal{S}^+ = Supp\,(\boldsymbol{d}_{\mathcal{I}})$. Define $\mathcal{Q} = (q_1, q_2, \ldots, q_{2T+1}, q_1)$, where $q_1, q_{2t+1} \in \mathcal{I}$ and $q_{2t} \in \mathcal{P}$ for all $t \in \{1,\ldots,T\}$. $\mathcal{Q}$ is an alternate cycle but for the $\{q_{2T+1}, q_1\}$ edge. Consider the two following cases:

(i) $\underline{q_1 \notin \mathcal{S}^+}$. Then $d^\star_{q_1} = 0$ and

$$\sum_{i\in\mathcal{Q}} x^\star_i = \sum_{i=2}^{2T+1} x^\star_{q_i} = \sum_{t=1}^{T} \left(x^\star_{q_{2t}} + x^\star_{q_{2t+1}} \right).$$

Every (q_{2t}, q_{2t+1}) is an edge of $\mathcal{G}$, so for every pair of columns $(\boldsymbol{A}_{\cdot 2t}, \boldsymbol{A}_{\cdot 2t+1})$ there exists a row i such that $A_{i(2t)} = A_{i(2t+1)} = 1$. The linear set partitioning constraint that corresponds to that conflict yields $x^\star_{q_{2t}} + x^\star_{q_{2t+1}} \leq 1$ for all $t \in \{1,\ldots T\}$. Hence, $\sum_{i\in\mathcal{Q}} x^\star_i \leq T = (|Q|-1)/2$ and $\boldsymbol{x}^\star$ does not violate $(\Gamma_{\mathcal{Q}})$.

(ii) $\underline{q_2 \notin \mathcal{S}^-}$. From $\boldsymbol{A}\boldsymbol{d}^\star = \boldsymbol{0}$ and $\boldsymbol{d}^\star_{\mathcal{I}} \geq \boldsymbol{0}$, necessarily $q_1, q_3 \notin \mathcal{S}^+$. Case (i) applies and this concludes the proof.

□

Proposition 9 suggests to distinguish between $\mathcal{S}^- - \mathcal{S}^+$ and $\mathcal{S}^+ - \mathcal{S}^+$ edges. Let $\mathcal{G}_B = (N_B, E_B)$ be a subgraph of $\mathcal{G}$ with $N_B = \mathcal{S}^- \cup \mathcal{S}^+ = Supp\,(\boldsymbol{d}^\star)$, and $\{i,j\} \in E_B$ iff $i \in \mathcal{S}^-$, $j \in \mathcal{S}^+$, and $\{i,j\} \in E$ (in conflict graph $\mathcal{G}$). $\mathcal{G}_B$ is a bipartite graph. In the case of the primal odd-cycle separation, the edges (and not the vertices) of the graph are weighted as $w_{ij} = 1 - x^\star_i - x^\star_j$ if $\{i,j\} \in E_B$. The weight of a path or a cycle is the sum of the weights of its edges. Given a cycle $\mathcal{Q}$, its weight is therefore $w(\mathcal{Q}) = |\mathcal{Q}| - 2\sum_{i\in\mathcal{Q}} x^\star_i$ and the corresponding odd-cycle inequality is violated by $\boldsymbol{x}^\star$ iff $w(\mathcal{Q}) < 1$. The primal odd-cycle separation procedure, CY_PSEP is given in Algorithm 3. This algorithm is usually fast because it consists of finding at most $|\mathcal{S}^-|$ shortest paths in a relatively small bipartite graph $\mathcal{G}_B$.

4.2 Algorithm

Algorithm 4 incorporates the cutting plane results discussed in the previous sections into Algorithm 1. The different (nonexhaustive) branching strategies that we use are presented in Section 5.

SEP_MAX is a parameter that represents the maximum number of separation problems to be solved consecutively. Explicit values chosen for INC_MAX and SEP_MAX, as well as the methods to determine the

Algorithm 3: CY_PSEP

Input: $\boldsymbol{d}^\star_{\mathcal{I}} \leftarrow$ an optimal solution of CP (fractional direction).
Output: $\mathcal{K}$, a set of primal odd-cycle cuts (empty if none exists)

$\mathcal{K} \leftarrow \emptyset$; $\boldsymbol{d}^\star \leftarrow \boldsymbol{T}\boldsymbol{d}^\star_{\mathcal{I}}$; $\mathcal{S}^- \leftarrow Supp\,(\boldsymbol{d}^\star_{\mathcal{P}})$; $\mathcal{S}^+ \leftarrow Supp\,(\boldsymbol{d}^\star_{\mathcal{I}})$; $\boldsymbol{x}^\star \leftarrow \boldsymbol{x}^0 + r(\boldsymbol{d}^\star)\boldsymbol{d}^\star$;
Build weighted graph $\mathcal{G}_B$ $(w_{ij} = 1 - x^\star_i - x^\star_j)$;
for $i, j \in \mathcal{S}^+$ *such that* $\{i, j\} \in E$ *($\mathcal{S}^+$–$\mathcal{S}^+$ conflict)* **do**
  $(q_1, q_2, \ldots, q_{2T}, q_{2T+1}) \leftarrow$ shortest path from i to j in $\mathcal{G}_B$ $(i = q_1,\ j = q_{2T+1})$;
  $\mathcal{Q} \leftarrow (q_1, q_2, \ldots, q_{2T}, q_{2T+1}, q_1)$ (odd-cycle);
  if $w(\mathcal{Q}) = w_{ij} + \sum_{k=1}^{2T+1} w_{q_k q_{k+1}} < 1$ **then**
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\Gamma_{\mathcal{Q}})\}$;
return $\mathcal{K}$;

set of variables to be fixed (line 33) are discussed in Section 5 before the numerical results are given. Different priority rules for choosing the separation algorithm (line 28) are studied and the corresponding results are displayed in the next section.

5 Numerical results

The results presented in this section empirically show the relevance of our theoretical results. We show that primal cuts indeed improve the performance of our algorithm and help foster integral solutions in ISUD.

5.1 Methodology

Instances. The instances presented here are those used by Zaghrouti et al. [40] (SPPAA01, VCS1200, and VCS1600) to which we added another instance from the OR-Library,[3] SPPAA04. Both SPPAA01 and SPPAA04 are small flight assignment problems (respectively 823 constraints and 8,904 variables, and 426 constraints and 7,195 constraints) with an average of 9 nonzero entries per column. VCS1200 is a medium-size bus driver scheduling problem (1,200 constraints, 133,000 variables), and VCS1600 is a large bus driver scheduling problem (1,600 constraints, 571,000 variables); both have an average of 40 nonzero entries per column. These numbers of nonzero entries per column are typical of aircrew and bus driver scheduling problems and do not vary much with the number of constraints. In scheduling instances, they typically correspond to the number of tasks to be completed by a worker in a given time span. The nonzeros correspond to the number of tasks per duty. The optimal solutions of the problems are known: SPPAA01 and SPPAA04 were solved with CPLEX 12.4,[4] and VCS1200 and VCS1600 were made with all columns generated during a column-generation process by GENCOL[5] whose optimal solution is known.

Initial solutions. In scheduling applications, a column generally represents a sequence of tasks performed by an employee. Define the *primal information* of a solution as the percentage of consecutive tasks in that solution that are also consecutive in the optimal schedule. In practice, this quantity is not known precisely a priori, but the following two observations hold for industrial bus driver or aircrew scheduling problems. First, crews do not often change vehicles during their duties, and their schedules therefore have many consecutive tasks in common with those of the vehicle routes. Second, when the schedule is updated, e.g., because of unforeseen events, the reoptimized schedule usually has many pieces in common with the original schedule (companies generally add penalties in the objective function to discourage changes). Thus, many consecutive tasks from the initial *paths* (vehicle routes or the original schedule) remain consecutive in the optimal schedule. In bus driver scheduling problems, the initial solution that follows the bus routes typically contains 90% of primal information (VCS1200, VCS1600). In aircrew scheduling, the figure is generally around 75% [40] (SPPAA01, SPPAA04). Therefore, we chose to perturb the (known) optimal solutions to generate initial

[3] http://people.brunel.ac.uk/~mastjjb/jeb/info.html
[4] CPLEX is freely available for academic and research purposes under the IBM academic initiative: http://www-03.ibm.com/ibm/university/academic
[5] GENCOL is a commercial software developed at the GERAD research center and now owned by the AD OPT company, a division of KRONOS.

Algorithm 4: `ISUD_CUTS`

Input: $\boldsymbol{x}^0$: a solution of SPP; INC_MAX: maximal incompatibility degree considered; SEP_MAX: maximal number of consecutive **P-SEP**;
Output: $\boldsymbol{x}^k$, a better solution of SPP.

$c \leftarrow 0$ (number of consecutive **P-SEP** solved);
$\iota \leftarrow 1$ (current maximum incompatibility degree);
`Pool` $\leftarrow \emptyset$ (pool of valid inequalities);
Compute $\mathcal{P}$ and $\mathcal{C}$ associated with $\boldsymbol{x}^0$; $k \leftarrow 0$;
while *true* **do**
  If necessary, update $\mathcal{P}$, $\mathcal{C}$, and $\mathcal{I}^\iota$ associated with $\boldsymbol{x}^k$;
  if $z^\star_{\mathrm{RP}} < 0$ **then**
    $\boldsymbol{\delta}^i \leftarrow$ an optimal solution of RP ($\bar{c}_i < 0$, $i \in \mathcal{C}^k$);
    $\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{x}^k + \boldsymbol{\delta}^i$; $k \leftarrow k+1$;
  else
    Build CP^ι anew (without cutting planes nor fixed variables);
    continue $\leftarrow$ true;
    while *continue = true* **do**
      If necessary, update $\mathcal{P}$, $\mathcal{C}$, and $\mathcal{I}^\iota$ associated with $\boldsymbol{x}^k$, and CP^ι;
      if $z^\star_{\mathrm{CP}^\iota} < 0$ **then**
        $\boldsymbol{d}^\star_{\mathcal{I}} \leftarrow$ an optimal solution of CP^ι; $\boldsymbol{d}^\star \leftarrow \boldsymbol{T}\boldsymbol{d}^\star_{\mathcal{I}}$;
        if $\boldsymbol{d}^\star_{\mathcal{I}}$ *is column-disjoint* **then**
          $\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{x}^k + r(\boldsymbol{d}^\star)\boldsymbol{d}^\star$;
          $k \leftarrow k+1$;
          continue $\leftarrow$ false;
        else
          $\boldsymbol{x}^\star \leftarrow \boldsymbol{x}^0 + r(\boldsymbol{d}^\star)\boldsymbol{d}^\star$;
          if $c <$ *SEP_MAX* **then**
            $c \leftarrow c+1$;
            if `Pool` *contains primal cuts that separate* $\boldsymbol{x}^\star$ *from* $\mathcal{F}_{\mathrm{SPP}}$ **then**
              Transfer all these cuts to CP^ι;
            else
              $\mathcal{K} \leftarrow$ set of primal cuts generated with `CL_PSEP` and/or `CY_PSEP`;
              if $\mathcal{K} \neq \emptyset$ **then**
                Transfer every cut in $\mathcal{K}$ to CP^ι;
                `Pool` $\leftarrow$ `Pool` $\cup\, \mathcal{K}$;
          else
            Fix at least one variable of CP^ι to zero;
            $c \leftarrow 0$;
      else
        if $\iota <$ *INC_MAX* **then**
          $\iota \leftarrow \iota + 1$;
          $c \leftarrow 0$;
        else
          return $\boldsymbol{x}^k$;

solutions that contain a similar level of primal information to that experienced in practice. These solutions are generally infeasible, so we assign the perturbed columns a high cost, as is done in companies for the schedules that follow the bus/airplane routes. In our perturbation method, the input parameter π is the percentage of columns of the optimal solution that will appear in the new initial solution. For SPPAA04 and SPPAA01, we generated initial solutions for $\pi = 10\%$, 15%, 20%, and 35%; for VCS1200 and VCS1600, we used $\pi = 20\%$, 35%, and 50%. These parameters were chosen so that the resulting primal information (given in Table 1) is consistent with the typical values. The initial gaps range from 50% to 80%, depending on the instance.

Table 1: Percentage of primal information in the initial solutions of the benchmark.

			Primal Information				
Instance	m	n	$\pi = 10\%$	$\pi = 15\%$	$\pi = 20\%$	$\pi = 35\%$	$\pi = 50\%$
SPPAA01	803	8,904	71.5	75.6	80.0	-	-
SPPAA04	423	7,195	-	64.0	70.1	78.5	-
VCS1200	1,200	133,000	-	-	87.6	91.1	93.9
VCS1600	1,600	570,000	-	-	86.8	90.9	93.9

Cutting Planes Strategies. The tests were conducted for the following cutting planes strategies:

NONE: Without primal cuts;
CLIQUE: With primal clique cuts only;
CYCLE: With primal odd-cycle cuts only;
BOTH: Both aforementioned cut types are separated at every **P-SEP** step;
PRIO: Primal odd-cycle cuts are separated only if no primal clique cut is found.

Moreover, another parameter is included, which is the maximum number of separation problems solved before using another technique (such as branching). We analyzed the results for different values ranging from 40 to 40,000 (virtually infinite). In a very large majority of the cases, either no cut could be found before the 40^{th} **P-SEP** was solved, or the cutting planes yield an integral direction in less than 40 **P-SEP**. Hence, we fixed the maximum number of consecutive **P-SEP** to 40 and only present these results here.

Branching Strategies. We propose four different nonexhaustive branching techniques to improve the performance of the algorithm. They correspond to the decision made when no primal cut is found that cuts the current fractional direction.

NOBR: stop the algorithm;
LAST: all variables of the last fractional direction found are set to zero in CP until an augmentation is performed;
FIRST: all variables of the first direction found since the last augmentation or the last branching are set to zero in CP until an augmentation is performed;
COVER: a subproblem is solved to determine a small subset of $\mathcal{I}$ such that the support of each fractional solution found since the last augmentation or the last branching contains at least one element of this set. These variables are set to zero in CP, so that all the aforementioned fractional solutions become infeasible.

All these techniques were experimented, but we present detailed results for NOBR and FIRST only. The performance for the other two branching strategies are exposed more briefly. The variation in the results is not significant enough to make a detailed presentation of all four aforementioned strategies.

5.2 Results

All the tests were performed on a Linux PC with processors of 3.4 GHz. For each problem and each perturbation parameter π, 10 different instances (different initial solutions and corresponding artificial columns) are generated. Column ALGO indicates the cutting strategy used; BEST is the best of the four on each instance, i.e., the one with the smallest gap, and in the event of a tie, the shortest computational time to reach the best solution. All times are given in seconds.

The commercial solver CPLEX was tested on all the instances of the benchmark, with the initial solutions given as MIP-start. Instances SPPAA01 and SPPAA04 are solved to optimality in an average of 22.1 and 14.4 seconds, respectively. Within a time limit of one hour, provided the initial solutions as a warm start, CPLEX only slightly improves them on the instance VCS1200, never lowering the gap under 14% (average gap of

49.7%). Within that same time limit, it never improves any of the initial solutions given for VCS1600 at all (the average gap remains 73% as for the initial solutions). Note that, if the initial solution is not given, CPLEX only finds a feasible solution for 1 of 30 instances for both VCS1200 and VCS1600 within one hour, and that solution is of comparable cost to that of our initial solutions. Here, one should consider SPPAA01 and SPPAA04 as benchmark instances used for a detailed analysis of our algorithm and its behavior, and VCS1200 and VCS1600 as practical instances that we aim to solve within a few minutes.

Tables 2 and 3 show the results of our algorithm for all instances generated from SPPAA01, with associated branching strategies NOBR and LAST, for parameters $INC\_MAX = 10$, $SEP\_MAX = 40$. Other values were tested and gave very similar results. Columns π and ALGO display the perturbation degree of the instances and the chosen separation strategy, respectively. The next three columns display the number of instances (out of 10) that were solved to optimality (0%), with a positive gap $\leq 2\%$, and with a gap $> 2\%$; the mean gap is given in column MEAN. Then, the overall computation time (t_{ISUD}), the time to reach the best solution obtained (t_{BEST}) and the average time per augmentation ($t_{\mathbf{AUG}}$) are displayed. The last columns contain the mean number of augmentations K performed to reach the best solution, and the mean size of $|\mathcal{S}| = Supp(\boldsymbol{d})$ for disjoint ($|\mathcal{S}|^D$) and nondisjoint ($|\mathcal{S}|^N$) directions. Note that, while the other mean values are computed over all executions of the algorithm, the mean number of augmentations K is based on the instances for which the final gap is $\leq 2\%$. Large gaps often mean a much smaller number of iterations, and taking these instances into account would distort that mean. Tables 4 and 5 show the results of the same experiments for instances generated from SPPAA04.

Consistently with our expectations and whatever the cutting planes or branching strategy, the primal information strongly influences the results of the algorithm. Namely, the higher the percentage of primal information is, the better the algorithm performs, both in terms of quality of the solution, and average running time. The branching strategies highlight common features and global differences between the various cutting techniques. First, the performance of the algorithms can be globally ranked from worst to best as follows: NONE $\preceq$ CYCLE $\preceq$ CLIQUE $\preceq$ PRIO $\preceq$ BOTH. Moreover, the execution time is significantly shorter for NONE and CYCLE than for the others. In the case of NONE, no primal separation problem is solved, and either no branching, or very simple fixing rules are applied, hence the high speed of the algorithm. In the case of CYCLE, the number of primal cycle cuts found is quite small, so the computation time is also smaller. The same holds for the time spent to reach the best solution (t_{BEST}) and the time per augmentation ($t_{\mathbf{AUG}}$). Note

Table 2: Comparison of the performances of ISUD with branching NOBR for instance SPPAA01. Parameters: $INC\_MAX = 10$ and $SEP\_MAX = 40$.

		GAP				Time (sec.)			**AUG**		
π	ALGO	0%	$\leq 2\%$	$> 2\%$	MEAN	t_{ISUD}	t_{BEST}	$t_{\mathbf{AUG}}$	K	$\lvert\mathcal{S}\rvert^D$	$\lvert\mathcal{S}\rvert^N$
10%	NONE	2	2	6	213.6%	6.2	5.1	0.21	29.	4.5	101.
	CLIQUE	3	5	2	72.1%	12.5	8.6	0.28	33.	4.9	122.
	CYCLE	3	2	5	197.5%	8.0	5.6	0.22	31.	4.6	143.
	BOTH	3	3	4	145.8%	15.9	7.4	0.25	35.	4.6	142.
	PRIO	3	5	2	72.1%	16.1	9.3	0.30	33.	4.9	113.
	BEST	**3**	**5**	**2**	**72.1%**	**11.7**	**7.9**	-	-	-	-
15%	NONE	2	3	5	134.1%	6.7	5.3	0.17	37.	3.7	48.
	CLIQUE	5	3	2	70.5%	10.4	6.7	0.18	39.	3.9	70.
	CYCLE	3	4	3	102.1%	8.0	5.9	0.18	37.	3.8	70.
	BOTH	5	4	1	41.4%	15.7	7.3	0.20	39.	4.0	95.
	PRIO	5	4	1	41.4%	15.6	7.4	0.20	39.	4.0	98.
	BEST	**5**	**4**	**1**	**41.4%**	**9.2**	**6.7**	-	-	-	-
20%	NONE	7	2	1	21.3%	7.6	6.0	0.17	37.	3.4	32.
	CLIQUE	5	5	0	0.2%	11.0	6.9	0.17	40.	3.4	57.
	CYCLE	6	3	1	40.7%	7.7	6.0	0.16	39.	3.4	43.
	BOTH	6	4	0	0.2%	15.0	7.0	0.18	40.	3.4	76.
	PRIO	6	4	0	0.2%	13.3	6.9	0.17	40.	3.4	74.
	BEST	**9**	**1**	**0**	**0.0%**	**10.4**	**6.3**	-	-	-	-

Table 3: Comparison of the performances of ISUD with branching LAST for instance SPPAA01. Parameters: $INC\_MAX = 10$ and $SEP\_MAX = 40$.

		GAP				Time (sec.)			**AUG**		
π	ALGO	0%	$\leq 2\%$	$> 2\%$	MEAN	t_{ISUD}	t_{BEST}	$t_{\mathbf{AUG}}$	K	$\lvert\mathcal{S}\rvert^D$	$\lvert\mathcal{S}\rvert^N$
10%	NONE	3	3	4	141.3%	11.4	8.7	0.27	38.	4.4	112.
	CLIQUE	3	3	4	80.2%	52.6	33.0	0.92	39.	4.7	198.
	CYCLE	3	2	5	139.0%	18.0	12.0	0.35	37.	4.3	151.
	BOTH	4	1	5	110.8%	76.2	32.2	0.91	40.	4.6	208.
	PRIO	5	1	4	79.1%	54.6	24.1	0.68	39.	4.7	177.
	BEST	**7**	**1**	**2**	**70.4%**	**33.2**	**19.4**	-	-	-	-
15%	NONE	2	4	4	30.9%	14.9	12.2	0.30	44.	4.0	87.
	CLIQUE	7	2	1	29.8%	28.2	9.6	0.24	41.	4.0	142.
	CYCLE	5	3	2	57.2%	16.0	10.6	0.27	41.	3.9	129.
	BOTH	8	2	0	0.0%	30.7	16.3	0.39	42.	4.1	111.
	PRIO	9	1	0	0.0%	34.6	17.6	0.43	41.	4.1	120.
	BEST	**9**	**1**	**0**	**0.0%**	**25.0**	**14.8**	-	-	-	-
20%	NONE	7	2	1	20.1%	9.9	7.0	0.17	42.	3.3	67.
	CLIQUE	6	4	0	0.0%	22.2	9.2	0.21	43.	3.5	104.
	CYCLE	8	2	0	0.1%	12.7	8.5	0.20	42.	3.4	86.
	BOTH	9	1	0	0.0%	23.6	9.8	0.22	45.	3.6	91.
	PRIO	9	1	0	0.0%	24.4	9.7	0.21	45.	3.6	102.
	BEST	**9**	**1**	**0**	**0.0%**	**16.1**	**9.1**	-	-	-	-

Table 4: Comparison of the performances of ISUD with branching NOBR for instance SPPAA04. Parameters: $INC\_MAX = 10$ and $SEP\_MAX = 40$.

		GAP				Time (sec.)			**AUG**		
π	ALGO	0%	$\leq 2\%$	$> 2\%$	MEAN	t_{ISUD}	t_{BEST}	$t_{\mathbf{AUG}}$	K	$\lvert\mathcal{S}\rvert^D$	$\lvert\mathcal{S}\rvert^N$
15%	NONE	1	0	9	518.8%	2.6	1.8	0.10	29.	3.4	86.
	CLIQUE	3	2	5	199.1%	8.1	3.3	0.14	26.	4.0	159.
	CYCLE	2	1	7	388.9%	3.7	2.3	0.12	26.	3.9	102.
	BOTH	5	2	3	160.8%	12.4	4.1	0.16	30.	4.1	154.
	PRIO	5	2	3	160.8%	9.8	4.0	0.16	30.	4.1	148.
	BEST	**5**	**2**	**3**	**160.8%**	**6.9**	**3.8**	-	-	-	-
20%	NONE	4	0	6	202.5%	3.2	2.2	0.10	28.	3.4	60.
	CLIQUE	5	2	3	83.0%	6.6	3.0	0.12	28.	3.7	138.
	CYCLE	4	0	6	202.5%	3.5	2.2	0.10	28.	3.4	93.
	BOTH	7	2	1	0.7%	11.6	3.7	0.14	27.	3.8	150.
	PRIO	7	2	1	0.7%	9.1	3.6	0.13	27.	3.8	143.
	BEST	**8**	**1**	**1**	**0.6%**	**5.9**	**3.2**	-	-	-	-
35%	NONE	9	0	1	0.6%	3.3	2.2	0.9	24.	3.0	73.
	CLIQUE	9	0	1	0.6%	7.1	2.2	0.9	25.	2.9	153.
	CYCLE	9	0	1	0.6%	3.8	2.2	0.9	25.	2.9	117.
	BOTH	9	0	1	0.6%	10.9	2.2	0.9	25.	2.9	162.
	PRIO	9	0	1	0.6%	8.1	2.2	0.9	25.	2.9	155.
	BEST	**9**	**0**	**1**	**0.6%**	**4.2**	**2.1**	-	-	-	-

that the difference is much bigger for t_{ISUD} than for t_{BEST} because the time spent at the optimal solution to generate cutting planes is important. The number of augmentation steps is higher for the algorithms that generate more primal cuts (column K). Generally, cutting planes allow the algorithm to find more disjoint solutions within the same phase, hence yielding a larger number of steps for each incompatibility degree. The algorithms using cuts tend to generate disjoint and nondisjoint combinations of larger size (columns $|\mathcal{S}|^D$ and $|\mathcal{S}|^N$). Cutting planes tend to make the problem more complex, add constraints, and the size of the support

Table 5: Comparison of the performances of ISUD with branching LAST for instance SPPAA04. Parameters: $INC\_MAX = 10$ and $SEP\_MAX = 40$.

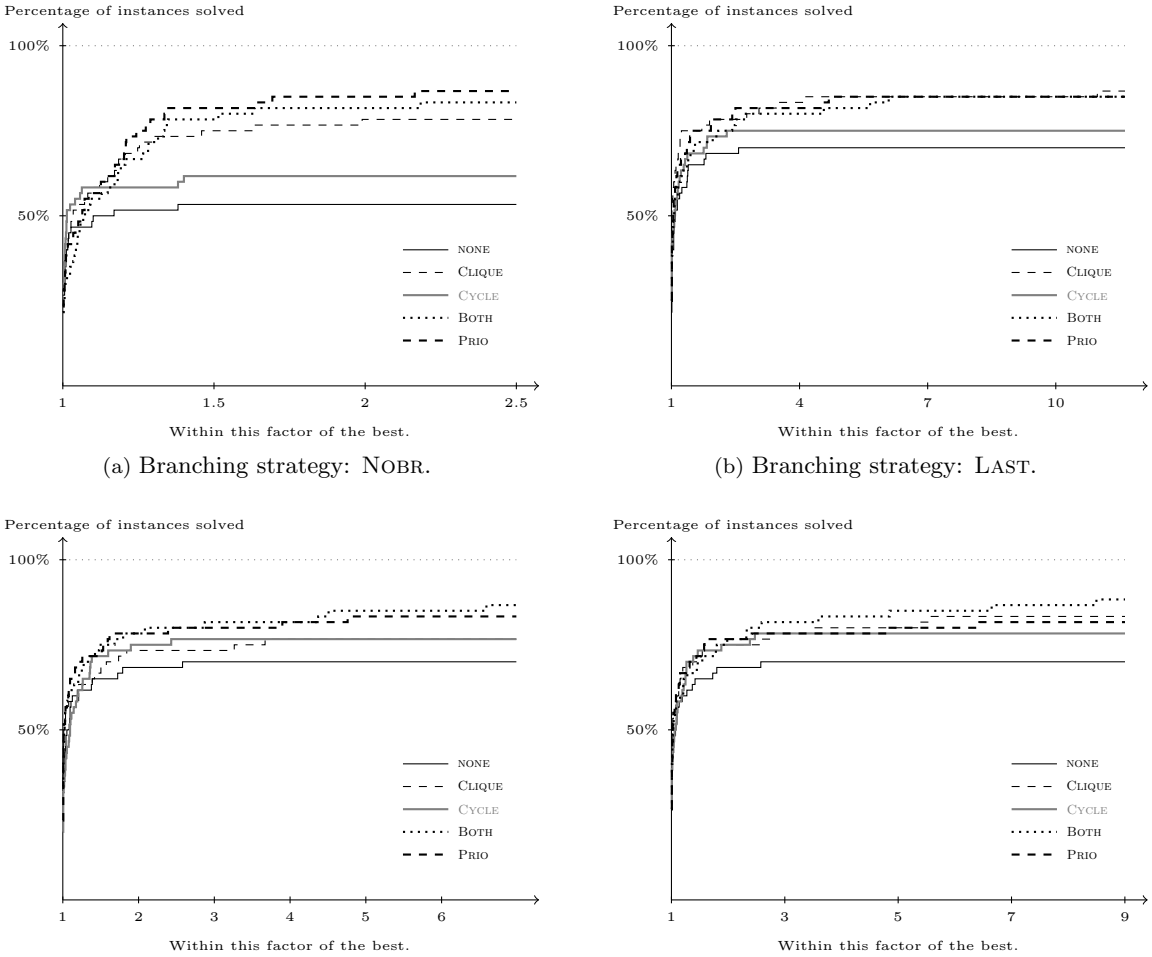
		GAP				Time (sec.)			**AUG**		
π	ALGO	0%	$\leq 2\%$	$> 2\%$	MEAN	t_{ISUD}	t_{BEST}	$t_{\mathbf{AUG}}$	K	$\lvert\mathcal{S}\rvert^D$	$\lvert\mathcal{S}\rvert^N$
15%	NONE	3	3	4	242.3%	6.2	4.9	0.18	31.	4.0	72.
	CLIQUE	4	3	3	189.3%	34.9	20.5	0.72	31.	4.1	180.
	CYCLE	3	3	4	211.5%	7.2	4.9	0.18	30.	3.9	102.
	BOTH	5	2	3	157.1%	34.3	13.9	0.48	31.	4.1	177.
	PRIO	5	2	3	157.1%	24.3	10.9	0.38	31.	4.1	168.
	BEST	**6**	**1**	**3**	**151.7%**	**24.3**	**15.0**	-	-	-	-
20%	NONE	3	2	5	224.3%	5.4	3.6	0.13	33.	3.4	76.
	CLIQUE	7	3	0	0.2%	15.6	9.7	0.30	32.	4.0	146.
	CYCLE	3	3	4	215.2%	9.0	5.2	0.18	35.	3.4	110.
	BOTH	9	0	1	48.3%	28.7	10.8	0.35	32.	3.9	150.
	PRIO	7	1	2	46.9%	27.8	14.0	0.47	31.	3.8	142.
	BEST	**10**	**0**	**0**	**0.0%**	**15.1**	**7.9**	-	-	-	-
35%	NONE	9	1	0	0.0%	3.5	2.4	0.9	25.	2.9	65.
	CLIQUE	9	1	0	0.1%	8.9	3.0	0.11	27.	3.0	140.
	CYCLE	9	1	0	0.0%	4.3	2.5	0.10	25.	2.9	108.
	BOTH	10	0	0	0.0%	12.9	2.5	0.10	27.	2.9	158.
	PRIO	10	0	0	0.0%	9.8	2.5	0.9	26.	2.9	150.
	BEST	**10**	**0**	**0**	**0.0%**	**5.3**	**2.4**	-	-	-	-

of a basic solution of CP therefore increases with the number of cuts inserted, hence the larger size of the combinations.

Furthermore, the algorithm is significantly better if fixing variables is available. This is particularly true when no cut is applied. Both nonexhaustive branching and cutting planes improve the performance of the algorithm. If we analyze the performance over SPPAA01 for $\pi = 15\%$ (closest to real-life problems) we see that (1) when no branching is made, PRIO solves 5/10 problems to optimality (against 2/10 for NONE), and 9/10 within 2% of the optimum (5/10 for NONE); and (2) when variables are fixed, PRIO solves 9/10 problems to optimality (2/10 for NONE) and all of them within 2% of the optimum (6/10 for NONE). Cutting planes therefore solve 7 out of the 8 problems that ISUD did not solve previously, hence yielding an improvement of 87.5% over SPPAA01 for $\pi = 15\%$.

Figure 6 displays the four performance diagrams of the algorithms for branching strategies NOBR, LAST, FIRST and COVER, over all SPPAA04 and SPPAA01 instances (70). Here again, whatever the branching strategy, cutting planes allow to solve many more instances. However, the improvement factor is much higher when no variable is fixed (less that 50% of the instances are solved without cuts; against more than 80% for PRIO or BOTH separation strategies). Interestingly, the time-increase factor (x-axis) never exceeds 10, and is most of the time lower than 4. Hence, the addition of cutting planes does not slow down the algorithm too much. Detailed computing times for each of the two problems are displayed in Figures 7 and 8, respectively. The COVER branching strategy shows slightly better performances, but not significantly enough to draw any conclusion yet. Moreover, more instances are solved by ISUD faster than by CPLEX.

Tables 6 and 7 display specific characteristics concerning the cutting planes generated during the process. For each branching strategy (NOBR and LAST), the number of instances solved within 2% of the optimum is shown under label INST. The mean time spent in the primal separation and cut management process including the cut pool management is given (t_{SEP}), as well as the mean number of separation problems solved (n_{SEP}), the mean number of primal clique cuts (n_{CL}), and primal odd-cycle cuts (n_{CY}). We can see that the time spent in the separation process is much higher when branching is applied. Indeed, in our algorithm, between two branching stages, SEP_MAX separation problems are solved. When no branching is performed, at most one sequence of SEP_MAX separation problems are solved and in case of failure to find a new primal cut, the algorithm instantly stops. This can also be seen in the n_{SEP} column, for which the number of separation



(a) Branching strategy: Nobr.

(b) Branching strategy: Last.

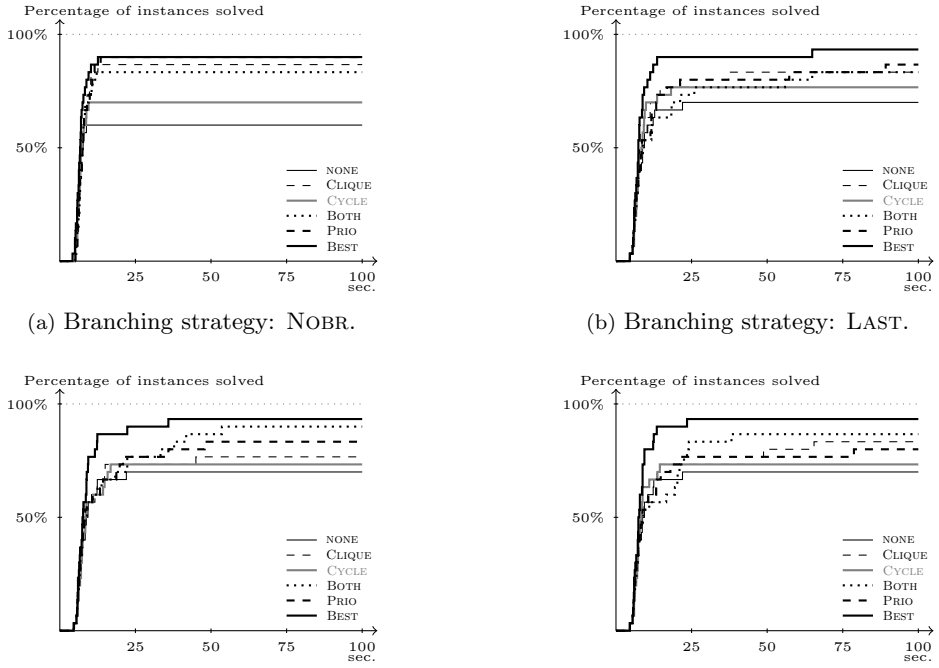
(c) Branching strategy: First.

(d) Branching strategy: Cover.

Figure 6: Performance diagrams over all sppaa04 and sppaa01 instances for the four branching strategies (Nobr (top-left), Last (top-right), First (bottom-left) and Cover (bottom-right). An instance is considered as solved if the gap is lower than 2%.

Table 6: Cutting planes behavior of Isud on sppaa01. Comparison of branching strategies Nobr and Last.

		inst		t_{Sep} (sec.)		mean					
						Nobr			Last		
gap	Algo	Nobr	Last	Nobr	Last	n_{Sep}	n_{Cl}	n_{Cy}	n_{Sep}	n_{Cl}	n_{Cy}
$\leq 2\%$	none	18	21	0.0	0.0	0	0	0	0	0	0
	Clique	26	25	0.5	2.8	56	240	0	164	853	0
	Cycle	21	23	0.3	0.7	12	0	28	44	0	125
	Both	25	25	1.7	7.1	74	289	95	159	808	233
	Prio	27	26	1.2	4.9	95	356	22	219	1073	48
$> 2\%$	none	12	9	0.0	0.0	0	0	0	0	0	0
	Clique	4	5	1.5	29.9	68	454	0	290	3383	0
	Cycle	9	7	1.1	6.8	21	0	122	90	0	424
	Both	5	5	7.1	51.8	59	653	202	241	2957	567
	Prio	3	4	4.8	27.3	141	998	79	336	3259	153



(a) Branching strategy: NOBR. (b) Branching strategy: LAST.

(c) Branching strategy: FIRST. (d) Branching strategy: COVER.

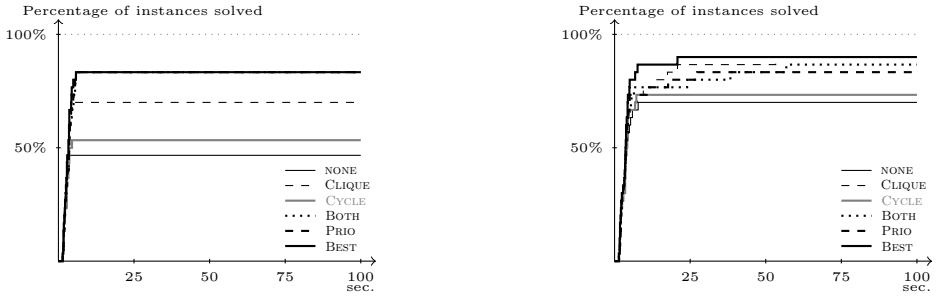
Figure 7: Percentage of instances solved over solution time for all SPPAA01 instances for the four branching strategies (NOBR (top-left), LAST (top-right), FIRST (bottom-left) and COVER (bottom-right). An instance is considered as solved if the gap is lower than 2%.

Table 7: Cutting planes behavior of ISUD on SPPAA04. Comparison of branching strategies NOBR and LAST.

		INST		t_{SEP} (sec.)		MEAN					
						NOBR			LAST		
GAP	ALGO	NOBR	LAST	NOBR	LAST	n_{SEP}	n_{CL}	n_{CY}	n_{SEP}	n_{CL}	n_{CY}
$\leq 2\%$	NONE	14	21	0.0	0.0	0	0	0	0	0	0
	CLIQUE	21	27	1.5	3.1	83	446	0	149	734	0
	CYCLE	16	22	0.3	0.5	15	0	52	29	0	100
	BOTH	25	26	5.2	8.4	91	438	103	138	665	165
	PRIO	25	25	2.2	3.2	106	480	31	144	658	44
$> 2\%$	NONE	16	9	0.0	0.0	0	0	0	0	0	0
	CLIQUE	9	3	1.6	34.9	58	366	0	540	3997	0
	CYCLE	14	8	0.4	3.3	17	0	39	101	0	514
	BOTH	5	4	5.2	41.5	74	426	88	375	2458	580
	PRIO	5	5	2.4	25.3	95	453	30	576	3033	216

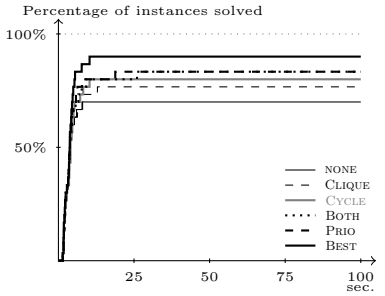
problems solved with branching is significantly higher than without branching. Moreover, the time t_{SEP}, and number of **P-SEP** n_{SEP} are significantly higher when the algorithm fails to find a good solution (gap $> 2\%$). This reflects the struggling of the algorithm to improve the solution and the associated running time increase. As it is often the case for the SPP, clique cuts are more efficient, and easier to find. Furthermore, as seen in Section 4.1, the requirements for an odd-cycle cut to be a primal cut are harder to meet (alternated cycles) than those that apply to clique cuts (only one vertex of the clique must be in the current solution); this also make them rarer.

Definition 4 A solution $\boldsymbol{x}^0$ is *ι-optimal* if it is optimal for the restriction of SPP to $\mathcal{P} \cup \mathcal{I}^\iota$, i.e., for the set of all at most ι-incompatible columns of $\boldsymbol{A}$ computed at $\boldsymbol{x}^0$.

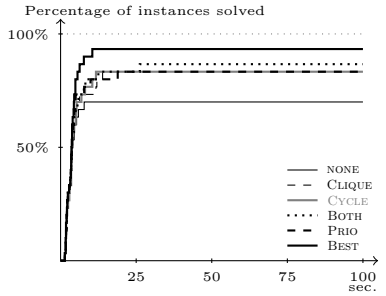


(a) Branching strategy: NOBR.

(b) Branching strategy: LAST.



(c) Branching strategy: FIRST.



(d) Branching strategy: COVER.

Figure 8: Percentage of instances solved over solution time for all SPPAA04 instances for the four branching strategies (NOBR (top-left), LAST (top-right), FIRST (bottom-left) and COVER (bottom-right). An instance is considered as solved if the gap is lower than 2%.

Detailed results of ISUD on VCS1200 and VCS1600 are displayed in Table 8 and 9. Branching did not change the results, hence the figures shown here correspond to the NOBR strategy and the time limit of half an hour was never reached. The first column shows the perturbation degree π, and the second the cutting plane strategy ALGO. The next four display the number of instances solved to optimality (0%), with a positive gap $\leq 2\%$ and with a gap $> 2\%$, as well as the mean gap (MEAN). Columns t_{ISUD} and t_{BEST} indicate the total running time and the time spent before reaching the best solution found, respectively. Then, the number of instances for which ι-optimality has been proved is given for $\iota = 7$ and $\iota = 8$, and finally the mean size of the disjoint ($|\mathcal{S}|^D$) and nondisjoint ($|\mathcal{S}|^N$) combination are shown.

In the experiments on VCS1200 and VCS1600, we chose a lower value for the maximum incompatibility number (INC_MAX= 8) than for the smaller problems; this choice is motivated by several reasons. First, it limits the running time of the algorithm. Second, this number is already high compared to what swap heuristics can consider. As explained in Section 3.1.2, the incompatibility degree of $\boldsymbol{A}_{\cdot j}$ is proportional to the number of sequences of consecutive tasks from $\boldsymbol{A}_{\cdot j}$ that are not performed consecutively in the current solution. Hence, it is a kind of measure of the primal distance between a column and the current solution. This number can be compared to the typical parameter of a swap heuristic that tries to swap parts of the current solution to form new individual schedules which is the number of times an individual schedule of the current solution may be split. Nonbasic columns for which $\iota = 8$ are made of at least 4 separate sequences of tasks performed consecutively in the current solution (there is no maximum number); in practice, this number usually ranges between 6 and 7. For an average of 40 nonzero entries per columns, this number therefore seems reasonable. It is in particular much higher than the typical number of splits allowed in the existing schedules in a swap heuristic, and the neighborhood explored by our algorithm is hence significantly wider than that of a swap heuristic.

Furthermore, the numerical results show that, even though primal cuts yield no improvement, they prove the ι-optimality of the solution in many cases, i.e., they show that the solution returned by ISUD is optimal for the restriction of SPP to the set of at most ι-incompatible columns ($\mathcal{I}^\iota$). This is especially true in the

Table 8: Results for instance VCS1200. Parameters: $INC\_MAX = 8$ and $SEP\_MAX = 40$; Branching NOBR.

		GAP				Time (sec.)		ι-OPT		**AUG**	
π	ALGO	0%	$\leq 2\%$	$> 2\%$	MEAN	t_{ISUD}	t_{BEST}	7-OPT	8-OPT	$\lvert\mathcal{S}\rvert^D$	$\lvert\mathcal{S}\rvert^N$
20%	NONE	4	0	6	20.3%	53	46	3	2	2.5	198
	CLIQUE	4	0	6	20.3%	112	46	8	2	2.5	480
	CYCLE	4	0	6	20.3%	55	46	3	2	2.5	213
	BOTH	4	0	6	20.3%	164	46	8	2	2.5	476
	PRIO	4	0	6	20.3%	112	46	8	2	2.5	480
35%	NONE	8	0	2	2.3%	50	43	0	0	2.3	151
	CLIQUE	8	0	2	2.3%	142	55	8	0	2.3	361
	CYCLE	8	0	2	2.3%	53	43	0	0	2.3	176
	BOTH	8	0	2	2.3%	166	49	8	0	2.3	371
	PRIO	8	0	2	2.3%	142	55	8	0	2.3	361
50%	NONE	10	0	0	0.0%	37	28	0	0	2.2	132
	CLIQUE	10	0	0	0.0%	81	29	10	0	2.2	315
	CYCLE	10	0	0	0.0%	39	29	0	0	2.2	156
	BOTH	10	0	0	0.0%	130	30	10	0	2.2	354
	PRIO	10	0	0	0.0%	81	29	10	0	2.2	315

Table 9: Results for instance VCS1600. Parameters: $INC\_MAX = 8$ and $SEP\_MAX = 40$; Branching NOBR.

		GAP				Time (sec.)		ι-OPT		**AUG**	
π	ALGO	0%	$\leq 2\%$	$> 2\%$	MEAN	t_{ISUD}	t_{BEST}	7-OPT	8-OPT	$\lvert\mathcal{S}\rvert^D$	$\lvert\mathcal{S}\rvert^N$
20%	NONE	5	0	5	16.5%	462	324	6	6	2.8	528
	CLIQUE	5	0	5	16.5%	646	325	6	6	2.8	831
	CYCLE	5	0	5	16.5%	465	324	6	6	2.8	534
	BOTH	5	0	5	16.5%	723	355	6	6	2.8	806
	PRIO	5	0	5	16.5%	647	325	6	6	2.8	831
35%	NONE	6	0	4	5.3%	433	244	7	6	2.4	519
	CLIQUE	6	0	4	5.3%	750	244	9	6	2.4	753
	CYCLE	6	0	4	5.3%	459	244	7	6	2.4	539
	BOTH	6	0	4	5.3%	813	244	9	6	2.4	761
	PRIO	6	0	4	5.3%	750	244	9	6	2.4	753
50%	NONE	8	0	2	2.4%	404	156	9	8	2.2	517
	CLIQUE	8	0	2	2.4%	492	156	9	8	2.2	691
	CYCLE	8	0	2	2.4%	405	156	9	8	2.2	521
	BOTH	8	0	2	2.4%	521	156	9	8	2.2	693
	PRIO	8	0	2	2.4%	492	156	9	8	2.2	691

case of VCS1200, where 3 instances are proven 7-optimal without cutting planes, whereas 26 are with cutting planes.

Finally, the failure of the cutting planes to improve the results of ISUD on VCS1200 and VCS1600 can be partly explained by the nature of these instances. Although they describe real-world problems, they are in fact large instances obtained from column generation solution of bus drivers scheduling instances. All columns generated throughout the Branch-and-Price process are stored and put together to form a large scale instance, the solution of which is known. However, due to the intrinsic nature of that solution process, the density of the vertices of the resulting relaxed polyhedron ($\mathcal{F}_{\text{SPP}^{LR}}$) tends to be higher near the optimal solution. Hence, if a wrong decision is made in the beginning, there is a high probability of going to a region of the polyhedron where very few extreme integer neighbors, and thus integer directions, exist. Getting back to an area where the density of integer neighbors is higher, only by traveling alongside integer-to-integer edges, may then become extremely complicated. In the context of all-integer column generation, i.e., alternatively generate columns and improve the integer solution with ISUD as evoked in the introduction, this issue would

disappear, because the dynamically generated columns increase the density of extreme points around the current solution, hence providing many more potential directions to the complementary problem.

6 Conclusions

In this work, we proposed a purely primal formulation of ISUD and introduced cutting planes in the complementary problem CP. We also presented the algorithm as a Matheuristic, i.e., the incompatibility degree defines a neighborhood of the current solution on which the augmentation problem is solved with integer programming techniques. The efficiency of linear programming and of our separation algorithms (Algorithms 2 and 3) allow us to explore a larger neighborhood than what a typical exchange heuristic would consider.

The work conducted on the mathematical formulation led us to characterize the set of cuts that can be transferred to CP as a nonempty subset of primal cuts tight at $\boldsymbol{x}^0$. We showed that, given a fractional direction $\boldsymbol{d}^\star$, a primal clique (or odd-cycle) cut exists if and only if there exists one that only involves variables of $Supp\,(\boldsymbol{d}^\star)$. Furthermore, in the case of primal odd-cycles, the separation over all variables is strictly equivalent to that over $Supp\,(\boldsymbol{d}^\star)$. Tests conducted over a benchmark of practical-like instances proved the potential of our method and highlight the importance of primal cutting planes in ISUD.

This work should be extended threefold. First, to gain better understanding of our algorithm, a larger benchmark is to be taken in consideration and a combination of cutting planes and other techniques (such as those proposed in [26] for instance) must be tested over that larger set of problems. Second, other cutting planes families should be taken in consideration, and the corresponding primal separation procedures must be developed. Last, the algorithm need to be extended to more general {0,1}-programs, and not only to SPP instances. This last point seems to be the most important if the method is to be used in practice, since supplementary constraints are often added to set partitioning problems. All these extensions are active research subjects and the present work will have extensions in a near future.

References

[1] Balas, E., Padberg, M.: On the set-covering problem: 2 – an algorithm for set partitioning. Operations Research 23(1), 74–90 (1975). DOI 10.1287/opre.23.1.74

[2] Baldacci, R., Mingozzi, A.: A unified exact method for solving different classes of vehicle routing problems. Mathematical Programming 120, 347–380 (2009)

[3] Ben-Israel, A., Charnes, A.: On some problems of diophantine programming. Cahiers du Centre d'Études de Recherche Opérationnelle 4, 215–280 (1962)

[4] Dakin, R.J.: A tree-search algorithm for mixed integer programming problems. The Computer Journal 8(3), 250–255 (1965). DOI 10.1093/comjnl/8.3.250. URL http://comjnl.oxfordjournals.org/content/8/3/250.abstract

[5] Desrosiers, J., Dumas, Y., Solomon, M.M., Soumis, F.: Time constrained routing and scheduling. Handbooks in operations research and management science 8, 35–139 (1995)

[6] Eisenbrand, F., Rinaldi, G., Ventura, P.: Primal separation for 0/1 polytopes. Mathematical Programming 95(3), 475–491 (2003)

[7] Elhallaoui, I., Metrane, A., Desaulniers, G., Soumis, F.: An improved primal simplex algorithm for degenerate linear programs. INFORMS Journal on Computing 23(4), 569–577 (2011)

[8] Elhallaoui, I., Metrane, A., Soumis, F., Desaulniers, G.: Multi-phase dynamic constraint aggregation for set partitioning type problems. Mathematical Programming 123(2), 345–370 (2010). DOI 10.1007/s10107-008-0254-5. URL http://dx.doi.org/10.1007/s10107-008-0254-5

[9] Garfinkel, R.S., Nemhauser, G.L.: The set-partitioning problem: Set covering with equality constraints. Operations Research 17(5), 848–856 (1969)

[10] Glover, F.: A new foundation for a simplified primal integer programming algorithm. Operations Research 16, 727–740 (1968)

[11] Goldberg, A.V., Tarjan, R.E.: Finding minimum-cost circulations by canceling negative cycles. Journal of the ACM 36(4), 873–886 (1989)

[12] Gomory, R.E.: Outline of an algorithm for integer solutions to linear program. Bulletin of the American Mathematical Society 64(5), 275–278 (1958)

[13] Gomory, R.E.: All-integer integer programming algorithm. Industrial Scheduling pp. 193–206 (1963)

[14] Haus, U.U., Köppe, M., Weismantel, R.: A primal all-integer algorithm based on irreducible solutions. Mathematical Programming 96(2), 205–246 (2003). DOI 10.1007/s10107-003-0384-8

[15] Kallio, M.J., Porteus, E.L.: A class of methods for linear programming. Mathematical Programming 14(1), 161–169 (1978)

[16] Letchford, A.N., Lodi, A.: Primal cutting plane algorithms revisited. Mathematical Methods of Operations Research 56(1), 67–81 (2002)

[17] Letchford, A.N., Lodi, A.: An augment-and-branch-and-cut framework for mixed 0-1 programming. In: M. Jünger, G. Reinelt, G. Rinaldi (eds.) Combinatorial Optimization – Eureka, You Shrink!, *Lecture Notes in Computer Science*, vol. 2570, pp. 119–133. Springer-Verlag Berlin Heidelberg (2003)

[18] Letchford, A.N., Lodi, A.: Primal separation algorithms. Quarterly Journal of the Belgian, French and Italian Operations Research Societies 1(3), 209–224 (2003)

[19] Metrane, A., Soumis, F., Elhallaoui, I.: Column generation decomposition with the degenerate constraints in the subproblem. European Journal of Operational Research 207(1), 37–44 (2010)

[20] Omer, J., Rosat, S., Raymond, V., Soumis, F.: Improved Primal Simplex: A More General Theoretical Framework and an Extended Experimental Analysis. Les Cahiers du GERAD G-2014-13, HEC Montréal, Canada (submitted to Informs Journal on Computing (2014)

[21] Östergård, P.R.J.: A new algorithm for the maximum-weight clique problem. Nordic Journal of Computing 8(4), 424–436 (2001)

[22] Padberg, M.W.: On the facial structure of set packing polyhedra. Mathematical Programming 5(1), 199–215 (1973). DOI 10.1007/BF01580121

[23] Rönnberg, E.: Contributions within two topics in integer programming : nurse scheduling and column generation. Ph.D. thesis, Linköping University (2012)

[24] Rönnberg, E., Larsson, T.: Column generation in the integral simplex method. European Journal of Operations Research 192(1), 333–342 (2009)

[25] Rönnberg, E., Larsson, T.: All-integer column generation for set partitioning: Basic principles and extensions. European Journal of Operational Research 233(3), 529–538 (2014). DOI http://dx.doi.org/10.1016/j.ejor.2013.08.036

[26] Rosat, S., Elhallaoui, I., Soumis, F., Chakour, D.: Influence of the normalization constraint on the integral simplex using decomposition. Les Cahiers du GERAD G-2014-99, HEC Montréal, Canada (submitted to Discrete Applied Mathematics) (2014)

[27] Rosat, S., Elhallaoui, I., Soumis, F., Lodi, A.: Integral simplex using decomposition with primal cuts. In: J. Gudmundsson, J. Katajainen (eds.) Experimental Algorithms, *Lecture Notes in Computer Science*, vol. 8504, pp. 22–33. Springer International Publishing (2014)

[28] Rozenknop, A., Wolfler Calvo, R., Alfandari, L., Chemla, D., Létocart, L.: Solving the electricity production planning problem by a column generation based heuristic. Journal of Scheduling 16(6), 585–604 (2013)

[29] Salkin, H.M., Koncal, R.D.: Set covering by an all-integer algorithm: Computational experience. Journal of the ACM 20(2), 189–193 (1973)

[30] Saxena, A.: Set-partitioning via integral simplex method. Unpublished manuscript, OR Group, Carnegie-Mellon University, Pittsburgh (2003)

[31] Schulz, A.S., Weismantel, R., Ziegler, G.M.: 0/1-integer programming: Optimization and augmentation are equivalent. In: P. Spirakis (ed.) ESA '95, *LNCS*, vol. 979, pp. 473–483. Springer Berlin Heidelberg (1995). DOI 10.1007/3-540-60313-1_164

[32] Spille, B., Weismantel, R.: Primal integer programming. In: K. Aardal, G. Nemhauser, R. Weismantel (eds.) Discrete Optimization, *Handbooks in Operations Research and Management Science*, vol. 12, pp. 245–276. Elsevier, Amsterdam (2005)

[33] Stallmann, M.F., Brglez, F.: High-contrast algorithm behavior: Observation, hypothesis, and experimental design. In: Proceedings of the 2007 Workshop on Experimental Computer Science, ExpCS '07. ACM, New York, NY, USA (2007). DOI 10.1145/1281700.1281712

[34] Stojković, M., Soumis, F., Desrosiers, J.: The operational airline crew scheduling problem. Transportation Science 32(3), 232–245 (1998)

[35] Thompson, G.L.: An integral simplex algorithm for solving combinatorial optimization problems. Computational Optimization and Applications 22(3), 351–367 (2002)

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

[37] Trubin, V.: On a method of solution of integer linear programming problems of a special kind. Soviet Mathematics Doklady 10, 1544–1546 (1969)

[38] Young, R.D.: A primal (all-integer) integer programming algorithm. Journal of Research of the National Bureau of Standards–B. Mathematics and Mathematical Physics pp. 213–250 (1965)

[39] Young, R.D.: A simplified primal (all-integer) integer programming algorithm. Operations Research 16(4), 750–782 (1968)

[40] Zaghrouti, A., Soumis, F., El Hallaoui, I.: Integral simplex using decomposition for the set partitioning problem. Operations Research 62(2), 435–449 (2014)