Les Cahiers du GERAD ISSN: 0711–2440

Dynamic penalization of fractional directions in the integral simplex using decomposition: Application to aircrew scheduling

S. Rosat, F. Quesnel,
I. Elhallaoui, F. Soumis

G–2016–01

January 2016

Cette version est mise à votre disposition conformément à la politique de libre accès aux publications des organismes subventionnaires canadiens et québécois.

Avant de citer ce rapport, veuillez visiter notre site Web (https://www.gerad.ca/fr/papers/G-2016-01) afin de mettre à jour vos données de référence, s'il a été publié dans une revue scientifique.

This version is available to you under the open access policy of Canadian and Quebec funding agencies.

Before citing this report, please visit our website (https://www.gerad.ca/en/papers/G-2016-01) to update your reference data, if it has been published in a scientific journal.

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, 2016
– Bibliothèque et Archives Canada, 2016

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, 2016
– Library and Archives Canada, 2016

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

Dynamic penalization of fractional directions in the integral simplex using decomposition: Application to aircrew scheduling

Samuel Rosat
Frédéric Quesnel
Issmail Elhallaoui
François Soumis

GERAD & Polytechnique Montréal, Montréal (Québec) Canada, H3C 3A7

samuel.rosat@gerad.ca
frederic.quesnel@gerad.ca
issmail.elhallaoui@gerad.ca
francois.soumis@gerad.ca

January 2016

Les Cahiers du GERAD
G–2016–01

Copyright © 2016 GERAD

Abstract: To solve integer linear programs, primal algorithms follow an augmenting sequence of integer solutions leading to an optimal solution. In this work, we focus on a particular primal algorithm, the integral simplex using decomposition (ISUD). To find the next point, one solves a linear program to select an augmenting direction for the current point from a cone of feasible directions. To ensure that this linear program is bounded, a normalization constraint is added and the optimization is performed on a section of the cone. The solution of the linear program, i.e., the direction proposed by the algorithm, strongly depends on the chosen normalization weights, and so does the likelihood that the next solution is integer. We modify ISUD so that the normalization is dynamically updated whenever the direction leads to a fractional solution, to penalize that direction. We propose several update strategies, based on theoretical and experimental results. To prove the efficiency of our strategies, we show that our version of the algorithm yields better results than the former version and than classical branch-and-bound techniques on a benchmark of industrial aircrew scheduling instances. The benchmark that we propose here is, to the best of our knowledge, comparable to no other from the literature. It provides large-scale instances with up to 1,700 flights and 115,000 pairings, hence as many constraints and variables, and the instances are given in a set-partitioning form together with initial solutions that accurately mimic those of industrial applications. Our work shows the strong potential of primal algorithms for the crew scheduling problem, which is a key challenge for large airlines.

Acknowledgments: This work was supported by a Collaborative research and development grant from NSERC and Kronos Inc. (RDCPJ 477127-14).

1 Introduction

Crew pairing is a key challenge for large airlines: it is both financially significant and notably hard to solve. The fundamental challenge is the size of the instances: large airlines have grown continually since the very beginning of passenger air traffic, as a result of increases in passenger volumes and occasional mergers (Continental Airlines and United Airlines, Air France and KLM, etc.). The problem involves determining a set of pairings (where a pairing is a sequence of flights and layovers that starts and ends at the same base) that covers all the scheduled flights at minimal cost over the planning horizon. The standard solution techniques are all based on the branch-and-bound algorithm. First, the integrality constraints are relaxed and an optimal fractional solution is found. Second, a branching tree is explored until integrality is restored and the integrality gap reduced to a given threshold. The exploration of the tree involves solving numerous linear relaxations of slightly modified versions of the original problem. The main drawbacks of this strategy are as follows. The size of the tree, and therefore the solution time, grows exponentially with the size of the data. Furthermore, the method used to solve the linear relaxations, the simplex algorithm, performs poorly on degenerate instances. An instance is *degenerate* when too many of the constraints are simultaneously saturated or, equivalently, some variables in the simplex basis have the value zero in most of the basic solutions. Crew scheduling problems have a high proportion of so-called set-partitioning constraints, and these increase the degeneracy. Moreover, it might take a long time to find an integer solution.

Another approach for these problems (and for integer linear programming in general) uses a *primal*, or *augmentation*, algorithm: it starts from a known feasible solution and iteratively improves it until optimality is reached. At each step it solves an augmentation subproblem that either furnishes an improving direction or asserts that the current solution is optimal. In the former case, this direction is followed from the current solution to a strictly better one. The main drawbacks of this method are the need for an initial feasible solution and the need to ensure that the improved solution is still integer. In the case of aircrew scheduling, the determination of good initial solutions proves to be relatively easy (see Section 5). The advantages of this approach are that it takes advantage of existing solutions, which is not possible with branch-and-bound, and it can quickly improve the given solution. The former is particularly important since it allows the approach to be used both for planning and for reoptimization after unforeseen events, when the existing solution can be used as a starting point.

This paper is organized as follows. In Section 2, we give an overview of augmentation methods, particularly in the context of the set partitioning problem (SPP) where these algorithms can be based on simplex pivots and are hence called *integral simplex* methods. In Section 3, we describe a specific augmentation algorithm, the integral simplex using decomposition (ISUD), based on work by Zaghrouti et al. [24] and Rosat et al. [13]. In Section 4 we improve this method by dynamically modifying one of the constraints in the augmentation subproblem. Four modification strategies are proposed, some based on mathematical properties and others on experimental findings. The dynamic update aims to increase the likelihood that the augmentation subproblem returns an augmenting direction that leads to an integral solution; it is an extension of the work of Rosat et al. [13]. In Section 5, we propose a new benchmark of SPPs generated from aircrew scheduling applications. This benchmark is an important contribution of the present work because it mimics as realistically as possible both the scheduling instances and the initial feasible solutions of real applications. To the best of our knowledge, no similar benchmark exists in the literature. In Section 6, we report numerical results that demonstrate the improvements obtained by our dynamic update of the constraints. Section 7 provides concluding remarks.

2 Integral simplex algorithms for the set partitioning problem

We assume that the reader is familiar with integer linear programming (see for example Schrijver [18]). Let ILP denote a generic integer linear program. A *primal* (or *augmentation*) algorithm is a method that, given an initial ILP solution, outputs a sequence of feasible (integer) ILP solutions of strictly augmenting cost such that the final solution is optimal for ILP. By *augmenting*, we mean increasing for a maximization problem and decreasing for a minimization problem. In this paper, we consider minimization but the ideas also

apply to maximization. Every primal method is based on the iterative solution of the *integral augmentation problem*, which can be stated as follows:

iAUG	Find an augmenting vector $\boldsymbol{z} \in \mathbb{Z}^n$ such that $(\boldsymbol{x} + \boldsymbol{z})$ is feasible for ILP and $\boldsymbol{z}$ is of negative cost, or assert that $\boldsymbol{x}$ is optimal for ILP.

Here $\mathbb{Z}^n$ is the set of integers, and $\boldsymbol{x}$ is a (known) solution of ILP. If an augmenting vector $\boldsymbol{z}$ is found, $\boldsymbol{x}' = (\boldsymbol{x} + \boldsymbol{z})$ becomes the next solution in the sequence, and **iAUG** is then solved again with $\boldsymbol{x}'$ as the initial solution. The process stops when $\boldsymbol{x}$ is determined to be optimal. Unfortunately, in general, the theoretical complexity of a single augmentation step is the same as that of solving the integer linear program [19], and integer linear programming is $\mathcal{NP}$-hard. However, in practice, **iAUG** or a good relaxation of **iAUG** is relatively easy to solve and often produces integral augmentations on specific problems, including the one we consider here. For a detailed review of primal algorithms, see Spille et al. [20].

We concentrate on a specific integer linear program, the SPP, because it is the core of scheduling models. The formulation is

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

where $\boldsymbol{A} \in \{0,1\}^{m \times n}$ is an $m \times n$ binary matrix, and $\boldsymbol{c} \in \mathbb{N}^n$ is the cost vector. Here $\boldsymbol{0}$ and $\boldsymbol{e}$ are vectors of zeroes and ones with size dictated by the context. Without loss of generality, we assume that $\boldsymbol{A}$ is full rank, contains no zero rows or columns, and has no identical rows or columns. The equalities $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{e}$ are called the linear constraints and $\boldsymbol{0} \leq \boldsymbol{x} \leq \boldsymbol{e}$ are the bound constraints. The set of all feasible solutions of SPP is denoted $\mathcal{F}_{\text{SPP}}$; the optimal value (or optimum) is $z^\star_{\text{SPP}}$; and any feasible solution $\boldsymbol{x}^\star \in \mathcal{F}_{\text{SPP}}$ such that $\boldsymbol{c}^T\boldsymbol{x}^\star = z^\star_{\text{SPP}}$ is an optimal solution of SPP. Because of the bounds ($\boldsymbol{0}$ and $\boldsymbol{e}$) and the integrality constraints, $\mathcal{F}_{\text{SPP}}$ contains only $\{0,1\}$-vectors, i.e., $\mathcal{F}_{\text{SPP}} \subseteq \{0,1\}^n$. Finally, the linear relaxation of SPP, denoted SPP^{LR}, is the linear program obtained by removing the integrality constraints. Its feasible domain is $\mathcal{F}_{\text{SPP}^{\text{LR}}}$ and its optimal value is $z^\star_{\text{SPP}^{\text{LR}}}$. If k augmentations have been performed, $\boldsymbol{x}^k$ designates a known integer solution that we seek to improve, and $\boldsymbol{x}^{k+1}$ is the next solution that we want to determine.

The SPP has specific features that allow the design of integral simplex methods. First, SPP is a $\{0,1\}$-program, so all the integer points of the domain of $\mathcal{F}_{\text{SPP}^{\text{LR}}}$ are extreme points of that polytope. Second, as shown by Trubin [23], it possesses the *quasi-integral* property, also called the *Trubin property*: every edge of the convex hull of the integer domain $\mathit{Conv}\,(\mathcal{F}_{\text{SPP}})$ is also an edge of $\mathcal{F}_{\text{SPP}^{\text{LR}}}$. These two observations show that from any initial solution there exists a finite sequence of strictly augmenting integer solutions $\left(\boldsymbol{x}^k\right)_k$ that reaches an optimal solution and has the property that any two consecutive points are adjacent in $\mathcal{F}_{\text{SPP}^{\text{LR}}}$. Two consecutive points are neighbors in $\mathcal{F}_{\text{SPP}^{\text{LR}}}$, so it is possible to go from one to the other by performing a sequence of simplex pivots. Moreover, it is always possible to find a sequence of pivots between two adjacent vertices such that only one of them is nondegenerate; otherwise they would not be adjacent. An integral simplex method is thus a primal algorithm in which an augmentation is performed through one or several simplex pivots, only one of which is nondegenerate.

SPPs have intrinsic degeneracy, and this is a challenge for primal algorithms, especially algorithms based on simplex pivots. Therefore, integral simplex implementations need techniques that are specifically designed to cope with degeneracy. One of the first algorithms to take degeneracy into account was that of Balas and Padberg [1]. Given an initial solution $\boldsymbol{x}^k$ of SPP, they generate a large number of augmenting directions and consider only those that lead to integral neighbors of $\boldsymbol{x}^k$. For an integer neighbor $\boldsymbol{x}^{k+1}$ of $\boldsymbol{x}^k$, let $\boldsymbol{d} = \boldsymbol{x}^{k+1} - \boldsymbol{x}^k$ be the corresponding direction. The positive support of $\boldsymbol{d}$, $\mathcal{Q}^+ = \{j | d_j > 0\}$, defines the set of indices of columns that should be pivoted into the basis to go from $\boldsymbol{x}^k$ to $\boldsymbol{x}^{k+1}$. In the terminology of Balas and Padberg [1], $\mathcal{Q}^+$ is *nondecomposable* if, for any strict subset $\tilde{\mathcal{Q}}^+ \subsetneq \mathcal{Q}^+$, performing pivots on the variables indexed by $\tilde{\mathcal{Q}}^+$ results in no change in the solution (degenerate pivots only). They prove that $\mathcal{Q}^+$ is nondecomposable if and only if $\boldsymbol{x}^{k+1}$ is a neighbor of $\boldsymbol{x}^k$. After restricting the set of directions to those that are integral, they choose the steepest and either perform the corresponding pivots if it is augmenting, or prove that $\boldsymbol{x}^k$ is optimal. If $\boldsymbol{x}^k$ is not optimal, $\boldsymbol{x}^{k+1}$ is different from $\boldsymbol{x}^k$ and has a strictly better cost. The main handicap of this method is that the number of integral neighbors of $\boldsymbol{x}^k$ and the number of potential directions that must be eliminated explode with the dimension of the search space (n).

Thompson [22] introduces a *local integral simplex method* (LISM) and a *global integral simplex method* (GLISM). LISM is based on a sequence of *pivots-on*-1, i.e., pivots on entries of the simplex tableau that take the value 1, and it is limited to augmenting pivots. For the SPP, given a basic solution $\boldsymbol{x}^k$, pivots-on-1 are exactly the simplex pivots that lead to a neighboring integer solution of $\boldsymbol{x}^k$. It may however be impossible to reach optimality by performing only augmenting pivots-on-1, and Thompson [22] therefore embeds LISM into the GLISM framework. In GLISM, a branching tree is explored, and its subproblems are solved using LISM. At each node of the tree, if LISM solves the subproblem to optimality, the branch no longer needs to be explored; otherwise, new branches are created and the process continues until the tree has been exhaustively explored. Thompson [22] shows promising numerical results on the Hoffman and Padberg benchmark [7]. Constraint propagation methods are used to speed up the solution of the linear relaxations, but the restriction to pivots-on-1 may lead to an explosion of the size of the branching tree. Although the total number of simplex pivots to reach optimality cannot exceed $2(n^2+n)^m$ (we do not doubt the theoretical importance of that result), this bound is of limited interest in practice. For example, in aircrew scheduling, m is typically the number of flights (often over $1,000$) and n the number of pairings (usually over $10,000$). Saxena [16, 17] extends the work of Thompson, exploring the characterization of neighboring solutions and the corresponding bases, anti-cycling techniques, and cutting planes. His work is mainly theoretical.

Stallman and Brglez [21] present a comparative study of their augmentation algorithm for the set covering problem (a variation of the SPP in which the equality constraints are replaced by $\geq$ inequalities). Although they give no major theoretical improvements, their work is an excellent example of a comparison between an all-integer algorithm and a standard solver based on branch-and-bound (IBM CPLEX, in this case). It is one of the few studies that give numerical results for sizable instances (up to 2,900 constraints).

One of the most interesting recent studies of integral simplex algorithms is that carried out by Rönnberg and Larsson as part of Rönnberg's PhD [10]. They propose an all-integer column-generation scheme [11] that they apply to the nurse scheduling problem [12]. Their algorithm allows both pivots-on-1 and pivots-on-(-1). The latter yield no change in the solution, so the approach is vulnerable to degeneracy.

Another pivot-based method in the same spirit is the ISUD of Zaghrouti et al. [24]. It is a promising recent development because it is based on linear programming techniques that take advantage of degeneracy and it guarantees strict improvement at each iteration (see [4, 9]). To find the edge leading to the next point, one solves a linear program to select an augmenting direction for the current point from the cone of feasible directions. To ensure that this linear program is bounded (the directions could go to infinity), a normalization constraint is added and the optimization is performed on a section of the cone. Solving this program with the simplex algorithm guarantees that the augmenting direction returned corresponds to an extreme ray of the cone and therefore to an edge of $\mathcal{F}_{\text{SPP}^{\text{LR}}}$. The solution of this linear program strongly depends on the chosen normalization weights, and so does the likelihood that the next solution is integer. Zaghrouti et al. [24] report promising results for medium aircrew scheduling and large bus driver scheduling instances. They consider all the weights to be equal to 1 in the normalization constraint; they therefore call it the *convexity* constraint. This linear program usually produces integral augmentations; when it does not, branching is necessary. Rosat et al. [13] strengthen the theoretical framework of ISUD, introduce a generic normalization constraint, explore the theoretical properties of some specific constraints, and discuss the practical design of the normalization constraint. They report computational results that improve on those of the original implementation. They also compare different normalization strategies and highlight their influence on the behavior of the algorithm. The present work focuses on a dynamic update of the normalization constraint to penalize fractional directions and increase the likelihood that the edge leads to an integer neighbor of $\boldsymbol{x}^k$.

3 The integral simplex using decomposition

In this section, we present the ISUD algorithm as it is described in Rosat et al. [13] (i.e., in the case of a generic normalization constraint) and discuss some of the main theoretical results.

3.1 Notation

We first introduce some notation, similar to that of [13]. For any polyhedron P, $Ext\,(P)$ is the set of its extreme points. We use lower-case bold symbols for column vectors and upper-case bold symbols for matrices. Given subsets $\mathcal{I} \subseteq \{1,\dots,m\}$ of the row indices and $\mathcal{J} \subseteq \{1,\dots,n\}$ of the column indices, the submatrix of $\boldsymbol{A}$ with rows indexed by $\mathcal{I}$ and columns indexed by $\mathcal{J}$ is denoted $\boldsymbol{A}_{\mathcal{I}\mathcal{J}}$. Similarly, $\boldsymbol{A}_{\mathcal{I}\cdot}$ is the set of rows of $\boldsymbol{A}$ indexed by $\mathcal{I}$, $\boldsymbol{A}_{\cdot\mathcal{J}}$ is the set of columns of $\boldsymbol{A}$ indexed by $\mathcal{J}$, and for any vector $\boldsymbol{v} \in \mathbb{R}^n$, $\boldsymbol{v}_{\mathcal{J}}$ is the subvector of all v_j, $j \in \mathcal{J}$. For any set $\mathcal{X}$, $\mathbb{1}_{\mathcal{X}}$ is the indicator vector of $\mathcal{X}$ with size dictated by the context, i.e., $[\mathbb{1}_{\mathcal{X}}]_j = 1$ if $j \in \mathcal{X}$, 0 otherwise. Finally, given any matrix $\boldsymbol{M}$, $\boldsymbol{M}^T$ is its transpose, and $\mathsf{Span}\,(\boldsymbol{M})$ is the linear span of its columns, also called the image of $\boldsymbol{M}$. Recall that, assuming that k augmentations have been performed, $\boldsymbol{x}^k$ designates a known integer solution that we seek to improve, and $\boldsymbol{x}^{k+1}$ is the next solution that we want to determine. The sets of indices of the positive and zero-valued variables of $\boldsymbol{x}^k$ are respectively denoted $\mathcal{P}^k$ and $\mathcal{Z}^k$ ($\mathcal{P}$ and $\mathcal{Z}$ when the context is clear). Hence, $\boldsymbol{x}^k_{\mathcal{P}} = \boldsymbol{e}$ and $\boldsymbol{x}^k_{\mathcal{Z}} = \boldsymbol{0}$. The set $\{1,\dots,n\}$ is thus partitioned into $(\mathcal{P},\mathcal{Z})$. Set $\mathcal{P}$ is called the *reduced basis* and has cardinality $p = |\mathcal{P}|$. Without loss of generality, the columns of $\boldsymbol{A}$ are assumed to be ordered such that $\mathcal{P} = \{1,\dots,p\}$. In this paper, the terms basis, basic, and nonbasic refer to this reduced basis $\mathcal{P}$.

Remark. The main differences between $\mathcal{P}$ and a classical simplex basis are that p may be smaller than m, and an extreme solution of SPP^{LR} is associated with a single reduced basis instead of potentially many simplex bases. When $p < m$, the set of columns of $\boldsymbol{A}_{\cdot\mathcal{P}}$ is not a basis of $\mathbb{R}^m$ in the algebraic sense but only a set of linearly independent vectors that spans a subspace of $\mathbb{R}^m$, $\mathsf{Span}\,(\boldsymbol{A}_{\cdot\mathcal{P}})$. In the classical simplex method, the reduced basis is completed with columns from $\boldsymbol{A}$, forming a basis of $\mathbb{R}^m$. Entering a column of $\boldsymbol{A}$ of negative reduced cost that is in $\mathsf{Span}\,(\boldsymbol{A}_{\cdot\mathcal{P}})$ into the basis produces a nondegenerate pivot that improves the solution. Such pivots are however not sufficient to reach optimality, and the columns of $\boldsymbol{A}$ that are not in $\mathsf{Span}\,(\boldsymbol{A}_{\cdot\mathcal{P}})$ must also be considered. To find a nondegenerate improvement with these columns, we can proceed as follows: we find a linear combination of these columns such that this combination (or *surrogate column*) is in $\mathsf{Span}\,(\boldsymbol{A}_{\cdot\mathcal{P}})$. If its reduced cost is negative, the solution is strictly improved by iteratively pivoting all the columns of the combination into the working basis. Further discussion of degeneracy and reduced bases can be found in the work of Omer et al. [9].

3.2 Fractional augmentation

Given $\boldsymbol{x}^k \in \mathcal{F}_{\text{SPP}}$ and a direction $\boldsymbol{d} \in \mathbb{R}^n$, $\boldsymbol{d}$ is called a *feasible* direction at $\boldsymbol{x}^k$ if it is possible to take a nonzero step in this direction while remaining feasible for SPP^{LR}, i.e., if $\exists \rho > 0 | \left(\boldsymbol{x}^k + \rho \boldsymbol{d}\right) \in \mathcal{F}_{\text{SPP}^{\text{LR}}}$. Moreover, it is called an *augmenting* direction if it has a negative cost ($\boldsymbol{c}^T\boldsymbol{d} < 0$). The largest ρ such that $\left(\boldsymbol{x}^k + \rho \boldsymbol{d}\right)$ is in $\mathcal{F}_{\text{SPP}^{\text{LR}}}$ is called the *maximal feasible step* in direction $\boldsymbol{d}$ from $\boldsymbol{x}^k$ and denoted $r(\boldsymbol{d})$ (or r or r^k when the context is clear). The set of all feasible directions at $\boldsymbol{x}^k$ is the cone $\{\boldsymbol{d} \in \mathbb{R}^n | \boldsymbol{A}\boldsymbol{d} = \boldsymbol{0}, \boldsymbol{d}_{\mathcal{P}} \leq \boldsymbol{0}, \boldsymbol{d}_{\mathcal{Z}} \geq \boldsymbol{0}\}$. The conditions $\boldsymbol{d}_{\mathcal{P}} \leq \boldsymbol{0}$ and $\boldsymbol{d}_{\mathcal{Z}} \geq \boldsymbol{0}$ are equivalent because all the entries of $\boldsymbol{A}$ are nonnegative. Furthermore, whenever this cone is not the singleton $\{\boldsymbol{0}\}$ (it is never a singleton in practice, since then $\mathcal{F}_{\text{SPP}}$ would also be a singleton), it is unbounded. Since $\boldsymbol{d} = \boldsymbol{0}$ is not an interesting direction if we seek to move from $\boldsymbol{x}^k$ to another solution, instead of considering the cone of the feasible directions, we will consider only a section of that cone. Let $\boldsymbol{w} \in \mathbb{R}^n$ be a normalization vector, and let

$$\Delta_{\boldsymbol{w}} = \left\{\boldsymbol{d} \in \mathbb{R}^n | \boldsymbol{A}\boldsymbol{d} = \boldsymbol{0}, \boldsymbol{w}^T\boldsymbol{d} = 1, \boldsymbol{d}_{\mathcal{Z}} \geq \boldsymbol{0}\right\} \tag{1}$$

be a section of the cone. A geometric description of $\Delta_{\boldsymbol{w}}$ is given in Figure 1. To ensure that $\Delta_{\boldsymbol{w}}$ is a proper section of the cone, we restrict our choice to $\boldsymbol{w}_{\mathcal{P}} \leq \boldsymbol{0}$ and $\boldsymbol{w}_{\mathcal{Z}} > \boldsymbol{0}$. The fractional augmentation problem (i.e., the problem of finding an augmented solution of the linear relaxation SPP^{LR} or asserting that $\boldsymbol{x}^k$ is optimal for SPP^{LR}) is therefore equivalent to determining whether or not the optimal value of the program, called the *maximum normalized augmentation*,

$$z^\star_{\text{MNA}^{\boldsymbol{w}}} = \min\left\{\boldsymbol{c}^T\boldsymbol{d} \,|\, \boldsymbol{d} \in \Delta_{\boldsymbol{w}}\right\} \tag{MNA$^{\boldsymbol{w}}$}$$

is negative. If it is negative, then any solution with a negative objective value is an augmenting feasible direction and the current solution can be (at least fractionally) improved. If it is nonnegative, no feasible direction at $\boldsymbol{x}^k$ is augmenting and $\boldsymbol{x}^k$ is therefore optimal for SPP^{LR} (and SPP).

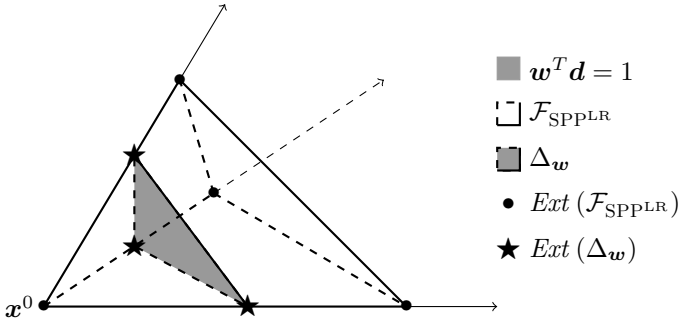

Figure 1: Geometric description of $\Delta_{\boldsymbol{w}}$. The cone of all feasible directions at $\boldsymbol{x}^0$ is defined by the three arrows. The grey area represents $\Delta_{\boldsymbol{w}}$, the set of normalized feasible directions.

Remark. The higher the weight associated with a variable, the less interesting the directions involving this variable. Increasing w_j pushes d_j toward zero and therefore tends to prevent $\boldsymbol{A}_{\cdot j}$ from entering the working basis. This can be seen as follows: let $\boldsymbol{w}^1$ and $\boldsymbol{w}^2$ be normalization vectors such that $\boldsymbol{w}^2$ is equal to $\boldsymbol{w}^1$ except for a given $j \in \mathcal{Z}$ for which $w_j^2 > w_j^1$. Given $\boldsymbol{d}$ in the cone of feasible directions, let $\boldsymbol{d}^1 = \boldsymbol{d}/(\boldsymbol{w}^1)^T\boldsymbol{d}$ and $\boldsymbol{d}^2 = \boldsymbol{d}/(\boldsymbol{w}^2)^T\boldsymbol{d}$ be the corresponding normalized directions in $\Delta_{\boldsymbol{w}^1}$ and $\Delta_{\boldsymbol{w}^2}$. If $d_j = 0$, the costs of these directions are equal, whereas if $d_j > 0$, $\boldsymbol{c}^T\boldsymbol{d}^1 < \boldsymbol{c}^T\boldsymbol{d}^2$. In the latter case, the normalized direction *appears* to be better if $w_1 < w_2$. Of course, the cost of a normalized direction is seen through the prism of $\boldsymbol{w}$. The determination of the maximal *normalized* augmenting direction ($\text{MNA}^{\boldsymbol{w}}$) provides no information on the length of the maximal step, so a direction that *appears* to be better than another one may not be better in practice.

3.3 Integral augmentation

In the previous section, we have solved the problem of a fractional augmentation, i.e., an augmentation within the linear relaxation polytope. To tackle the integral augmentation problem **iAUG**, we introduce further notation. A feasible direction $\boldsymbol{d} \in \mathbb{R}^n$ is called an *integral* direction if it leads to another integer solution of SPP, i.e., $\exists \rho > 0 | \left(\boldsymbol{x}^k + \rho \boldsymbol{d}\right) \in \mathcal{F}_{\text{SPP}}$. Proposition 1 below [13] gives a characterization of extreme integral directions.

Definition 1 *A set of indices $\mathcal{S} \subseteq \{1, \ldots, n\}$ is* column-disjoint *if no pair of columns of $\boldsymbol{A}_{\cdot\mathcal{S}}$ has a common nonzero entry, i.e., $\forall i, j \in \mathcal{S}, i \neq j, \forall k \in \{1, \ldots, m\}, A_{ki}A_{kj} = 0$.*

By extension, $\boldsymbol{A}_{\cdot\mathcal{S}}$ and $\boldsymbol{x}_{\mathcal{S}}$ are said to be column-disjoint if $\mathcal{S}$ is. The set of all integral directions is a subset of $\Delta_{\boldsymbol{w}}$ denoted $\Delta_{\boldsymbol{w}}^{int}$. Since $\boldsymbol{A}$ is binary, $\mathcal{S}$ is column-disjoint iff $\boldsymbol{A}_{\cdot\mathcal{S}}$ is a set of orthogonal columns.

Proposition 1 (Proposition 7 of [13]) *Given $\boldsymbol{d}^\star \in Ext\,(\Delta_{\boldsymbol{w}})$, $\mathcal{S}^+ = Supp\,(\boldsymbol{d}^\star_{\mathcal{Z}})$, and $\mathcal{S}^- = Supp\,(\boldsymbol{d}^\star_{\mathcal{P}})$, ($\boldsymbol{d}^\star$ is an integral direction) $\Leftrightarrow$ ($\mathcal{S}^+$ is column-disjoint). For any such direction, $d_j^\star = -\delta_0$ if $j \in \mathcal{S}^-$, δ_0 if $j \in \mathcal{S}^+$, and 0 otherwise, where $\delta_0 = (\sum_{j\in\mathcal{S}^-\cup\mathcal{S}^+} |w_j|)^{-1} = (-\sum_{j\in\mathcal{S}^-} w_j + \sum_{j\in\mathcal{S}^+} w_j)^{-1}$. Moreover, the maximal step in that direction is $r = 1/\delta_0$.*

Because SPP is quasi-integral, the edges of $Conv\,(\mathcal{F}_{\text{SPP}})$ are edges of $\mathcal{F}_{\text{SPP}^{\text{LR}}}$, so $Ext\left(\Delta_{\boldsymbol{w}}^{int}\right) \subseteq Ext\,(\Delta_{\boldsymbol{w}})$. In practice, $\text{MNA}^{\boldsymbol{w}}$ is solved with the simplex algorithm, and the optimal solution belongs to $Ext\,(\Delta_{\boldsymbol{w}})$. By Proposition 1, it is sufficient to test whether the solution of $\text{MNA}^{\boldsymbol{w}}$ is column-disjoint to determine if it is integral. Furthermore, Rosat et al. [13] show that, given any extreme feasible direction $\bar{\boldsymbol{d}}$ of negative cost ($\boldsymbol{c}^T\boldsymbol{d} < 0$), there exists a weight vector $\boldsymbol{w}$ such that the optimal solution $\boldsymbol{d}^\star$ of $\text{MNA}^{\boldsymbol{w}}$ is positively proportional to $\bar{\boldsymbol{d}}$. More precisely, the normalized solution is $\boldsymbol{d}^\star = \bar{\boldsymbol{d}}/\boldsymbol{w}^T\bar{\boldsymbol{d}}$ and minimizes $\left(\boldsymbol{c}^T\boldsymbol{d}/\boldsymbol{w}^T\boldsymbol{d}\right)$ over the whole cone of feasible directions (and also over $\Delta_{\boldsymbol{w}}$). The solution of $\text{MNA}^{\boldsymbol{w}}$, i.e., the direction proposed

by the algorithm, thus strongly depends on the chosen normalization weights, and so does the likelihood that the next solution is integer. Rosat et al. [13] also give advice on how to determine good normalization weights and offer four strategies that we summarize here. Note that when we write $\boldsymbol{w}$ as the concatenation of two vectors $(\boldsymbol{u}, \boldsymbol{v})$, we mean that $\boldsymbol{w}_{\mathcal{P}} = \boldsymbol{u}$ and $\boldsymbol{w}_{\mathcal{Z}} = \boldsymbol{v}$.

1. MMA: Nonweighted normalization vector $\boldsymbol{w} = (-\boldsymbol{e}, \boldsymbol{e})$. The corresponding problem $\text{MNA}^{(-\boldsymbol{e},\boldsymbol{e})}$ is called the *maximum mean augmentation* problem. Theoretical results on the maximum number of augmentations to reach optimality have been given for this normalization constraint (see [20]).
2. MIMA: Nonweighted normalization vector that applies weights only to the nonbasic (or incoming) variables $\boldsymbol{w} = (\boldsymbol{0}, \boldsymbol{e})$. The corresponding problem $\text{MNA}^{(\boldsymbol{0},\boldsymbol{e})}$ is called the *minimum incoming mean augmentation* problem. This weight vector appears in the seminal work of Zaghrouti et al. [24]. It has interesting theoretical properties, particularly when used in the decomposition context (see [13]).
3. NORM: The weight w_j of every nonbasic variable d_j equals the number of nonzero entries of the corresponding column of $\boldsymbol{A}$, called the *norm* of the column and denoted n_j: $\boldsymbol{w} = (\boldsymbol{0}, \boldsymbol{n})$. The higher the weight, the less likely that the variable enters the basis. This weight vector therefore tends to prevent denser columns from entering the basis and thus fosters disjoint solutions.
4. DEG: The weight w_j of every nonbasic variable d_j equals its *incompatibility degree* w.r.t. the current working basis $\mathcal{P}$, denoted ι_j: $\boldsymbol{w} = (\boldsymbol{0}, \boldsymbol{\iota})$. The incompatibility degree is described in Section 3.4.

The reader may object that comparing the optimal values of $\text{MNA}^{\boldsymbol{w}}$ and $\text{MNA}^{\boldsymbol{w'}}$ for different normalization constraints does not make sense ($\text{MNA}^{\boldsymbol{w}}$ minimizes $\boldsymbol{c}^T\boldsymbol{d}/\boldsymbol{w}^T\boldsymbol{d}$ over the cone of feasible directions, while $\text{MNA}^{\boldsymbol{w'}}$ minimizes $\boldsymbol{c}^T\boldsymbol{d}/\boldsymbol{w'}^T\boldsymbol{d}$ over the same domain). For this reason, we define the mean scaled cost of a direction that will allow us to compare the quality of two directions and, in particular, of the optimal solutions of $\text{MNA}^{\boldsymbol{w}}$ and $\text{MNA}^{\boldsymbol{w'}}$ when $\boldsymbol{w} \neq \boldsymbol{w'}$. This will be important later.

Definition 2 *The* mean scaled cost *of direction* $\boldsymbol{d}$ *is the cost of its mean scaling* $\boldsymbol{d}^{\boldsymbol{e}} = \boldsymbol{d}/(-\boldsymbol{e}, \boldsymbol{e})^T\boldsymbol{d} = \boldsymbol{d}/(\sum_{j\in\{1,\dots,n\}} |d_j|)$ *(corresponding to the maximum mean augmentation* MMA *mentioned above).*

3.4 Incompatibility degree of a column

Let $\mathsf{V}_1 = \mathsf{Span}\,(\boldsymbol{A}_{\cdot\mathcal{P}})$ be the subspace of $\mathbb{R}^m$ spanned by $\boldsymbol{A}_{\cdot\mathcal{P}}$. By definition, the columns of $\boldsymbol{A}_{\cdot\mathcal{P}}$ are linearly independent, hence they form a basis $\boldsymbol{A}_{\cdot\mathcal{P}}$ of V_1. One can complete this basis with a set of $m-p$ independent vectors $\boldsymbol{B_2} = \left[\boldsymbol{v^{p+1}} \dots \boldsymbol{v^m}\right]$ and denote by V_2 the subspace spanned by these vectors. By definition, V_1 and V_2 are supplementary subspaces of $\mathbb{R}^m$ ($\mathbb{R}^m = \mathsf{V}_1 \oplus \mathsf{V}_2$). The matrix built by the concatenation of the aforementioned bases $\boldsymbol{B} = \left[\boldsymbol{A}_{\cdot\mathcal{P}} \quad \boldsymbol{B_2}\right]$ is nonsingular by construction; its inverse is denoted $\boldsymbol{B}^{-1}$. One can freely transform the linear equations that define $\Delta_{\boldsymbol{w}}$ by multiplying them by $\boldsymbol{B}^{-1}$, so $\boldsymbol{Ad} = \boldsymbol{0}$ becomes $\boldsymbol{B}^{-1}\boldsymbol{Ad} = \boldsymbol{0}$. Because the current solution $\boldsymbol{x}^k$ is integral and because of the nature of the SPP, one can reorder the rows of $\boldsymbol{A}$ such that

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I}_p & \boldsymbol{A}_{\mathcal{P}\mathcal{Z}} \\ \boldsymbol{0} & \boldsymbol{A}_{\bar{\mathcal{P}}\mathcal{Z}} \end{bmatrix}, \text{ and therefore } \bar{\boldsymbol{A}} = \boldsymbol{B}^{-1}\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I}_p & \bar{\boldsymbol{A}}_{\mathcal{P}\mathcal{Z}} \\ \boldsymbol{0} & \bar{\boldsymbol{A}}_{\bar{\mathcal{P}}\mathcal{Z}} \end{bmatrix}, \tag{2}$$

where $\mathcal{P} = \{1, \dots, p\}$ and $\bar{\mathcal{P}} = \{p+1, \dots, m\}$. For $j \in \{1, \dots, n\}$, the *incompatibility degree* ι_j of the column $\boldsymbol{A}_j$ w.r.t. $\mathcal{P}$ and $\boldsymbol{B}$ is the number of nonzero entries in the lower part of $\bar{\boldsymbol{A}}_{\cdot j}$, i.e., in $\bar{\boldsymbol{A}}_{\bar{\mathcal{P}}j}$. Column $\boldsymbol{A}_{\cdot j}$ is said to be ι_j-incompatible, and 0-incompatible columns are called *compatible* columns. As can be seen in Equation (2), all columns from the working basis $\boldsymbol{A}_{\cdot\mathcal{P}}$ are compatible. In practice, V_2 and $\boldsymbol{B_2}$ are chosen so that $\boldsymbol{B}^{-1}$ is easy to compute. Basis completion is discussed further in [2], and the reader is encouraged to read [6] for a global overview of the underlying vector space decomposition structure. To the best of our knowledge, however, this is the first time that Proposition 2 below has been stated in these terms. We think that this result provides interesting insight and clarifies the notion of incompatibility degree. The reader should note that the vector spaces V_1 and V_2 may not be orthogonal. For the proposition, let $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ respectively be the projection on V_1 parallel to V_2 and on V_2 parallel to V_1, i.e., for any vector $\boldsymbol{v} \in \mathbb{R}^m$, $\boldsymbol{v} = \boldsymbol{\pi}_1(\boldsymbol{v}) + \boldsymbol{\pi}_2(\boldsymbol{v})$, $\boldsymbol{\pi}_1(\boldsymbol{v}) \in \mathsf{V}_1$ and $\boldsymbol{\pi}_2(\boldsymbol{v}) \in \mathsf{V}_2$.

Proposition 2 *With the above definitions, the incompatibility degree of column $\boldsymbol{A}_{\cdot j}$ is the number of nonzero coefficients in the expression of $\boldsymbol{\pi}_2(\boldsymbol{v})$ as a linear combination of elements of the basis $\boldsymbol{B_2}$ of V_2. It is equal to the number of elements of $\boldsymbol{B_2}$ involved in the decomposition of $\boldsymbol{v}$ on basis $\boldsymbol{B} = \begin{bmatrix}\boldsymbol{A}_{\cdot\mathcal{P}} & \boldsymbol{B_2}\end{bmatrix}$.*

Proof. Let $j \in \{1, \ldots, n\}$. Because $\boldsymbol{B}$ is a basis of $\mathbb{R}^m$, there exists a single $\boldsymbol{\lambda} \in \mathbb{R}^m$ such that $\boldsymbol{A}_{\cdot j} = \sum_{i=1}^{p} \lambda_i \boldsymbol{A}_{\cdot i} + \sum_{i=p+1}^{m} \lambda_i \boldsymbol{v}_i$. By definition of $\boldsymbol{B}^{-1}$ and $\bar{\boldsymbol{A}}$,

$$\bar{\boldsymbol{A}}_{\cdot j} = \boldsymbol{B}^{-1}\boldsymbol{A}_{\cdot j} = \left(\sum_{i=1}^{p} \lambda_i \boldsymbol{B}^{-1}\boldsymbol{A}_{\cdot i} + \sum_{i=p+1}^{m} \lambda_i \boldsymbol{B}^{-1}\boldsymbol{v^i}\right) = \sum_{i=1}^{m} \lambda_i \boldsymbol{\epsilon}^i, \tag{3}$$

where for all $i \in \{1, \ldots, m\}$, $\boldsymbol{\epsilon}^i$ is the vector defined by $\epsilon^i_j = 1$ if $i = j$ and 0 otherwise. Thus, ι_j is the number of nonzero elements in $\boldsymbol{\lambda}_{\bar{\mathcal{P}}}$. Since $\boldsymbol{\pi}_1(\boldsymbol{v}) = \sum_{i=1}^{p} \lambda_i \boldsymbol{A}_{\cdot i}$ and $\boldsymbol{\pi}_2(\boldsymbol{v}) = \sum_{i=p+1}^{m} \lambda_i \boldsymbol{v^i}$, the proposition holds. □

Proposition 2 shows the paramount importance of the choice of the vectors $\boldsymbol{v}^{p+1}, \ldots, \boldsymbol{v}^m$ when completing the working basis $\boldsymbol{A}_{\cdot\mathcal{P}}$. For example, when we build a simplex basis, the choice is limited to columns of $\boldsymbol{A}_{\cdot\mathcal{Z}}$. Here, we extend that choice to any basis completion of $\mathbb{R}^m$. However, because $\boldsymbol{v^{p+1}}, \ldots, \boldsymbol{v^m}$ are chosen arbitrarily, there is no point in performing what could be called degenerate pivots. These would result in a change of the completion but no change in the working basis $\boldsymbol{A}_{\cdot\mathcal{P}}$. A change of the completion is exactly what we do not want to perform, since it entails the computation of the new inverse matrix $\boldsymbol{B}^{-1}$ (whereas the chosen completion should ease this computation) and changes all the incompatibility degrees. We make the following recommendations regarding the choice of the basis completion. First, it should make the computation of $\boldsymbol{B}^{-1}$ fast; if the matrix inversion is slow it will jeopardize the whole process. Second, the columns of $\boldsymbol{A}$ that are of interest in the pursuit of optimality as well as those that are likely to appear in disjoint combinations should be as sparse as possible in $\bar{\boldsymbol{A}}$. If these conditions hold, using incompatibility degrees as normalization weights (the DEG constraint in Section 3.3) and/or performing partial pricing on low-ι columns give the algorithm a twofold advantage. Such a partial pricing is in the spirit of the multi-phase strategy that Elhallaoui et al. [5] propose for the dynamic constraint aggregation algorithm. Our choice of basis completion yields incompatibility degrees as described in [14].

3.5 Pseudocode

The practical implementation of ISUD borrows some ideas from variable neighborhood search to speed up the algorithm (see Algorithm 1). The set of columns considered in the augmentation problem is not $\mathcal{Z}$, but the restriction $\mathcal{Z}_\iota$ of $\mathcal{Z}$ to at most ι-incompatible variables, i.e., variables that are ι'-incompatible for $\iota' \leq \iota$. Here ι is called the *phase number*. The corresponding restriction of $\text{MNA}^{\boldsymbol{w}}$ to variables in $\mathcal{P}$ and $\mathcal{Z}_\iota$ is denoted $\text{MNA}^{\boldsymbol{w}}_{\mathcal{Z}_\iota}$. We thus solve an augmentation problem in which at most ι-incompatible columns are allowed to enter the working basis. The algorithm stops either when the neighborhood comprises all the variables in $\mathcal{Z}$ or when a maximum phase number is reached (heuristic stopping variant, line 2 of Algorithm 1). For the sake of clarity, hereinafter we will consider solving the version of $\text{MNA}^{\boldsymbol{w}}$ in which all columns of $\mathcal{Z}$ are present. All the results can then be adapted by simply replacing $\mathcal{Z}$ by $\mathcal{Z}_\iota$. To increase the chance of finding integral augmenting directions, we implement a simple nonexhaustive branching (fixing) strategy. Whenever a non-column-disjoint augmenting direction $\boldsymbol{d}^\star$ is found, all the entering variables from $\boldsymbol{d}_\mathcal{Z}$ are set to zero and the augmentation problem is solved again. Variables fixed to zero are freed only when the phase number ι is increased.

3.6 Further remarks on algorithm

Remark. Zaghrouti et al. [24] show that $\Delta_{\boldsymbol{w}}$ can be projected into the space of $\boldsymbol{d}_\mathcal{Z}$ without loss of information. In that subspace, we need p fewer constraints and p fewer variables to describe the cone section, and we obtain the new linear constraints $\bar{\boldsymbol{A}}\boldsymbol{d}_\mathcal{Z} = 0$, costs $\bar{\boldsymbol{c}}$, and normalization weights $\bar{\boldsymbol{w}}$ from the original ones via an easy-to-determine linear transformation. This speeds up the solution process, and we therefore use this transformation in our implementation.

Algorithm 1: Integral Simplex Using Decomposition

Input: $\boldsymbol{x}^0$, a solution of SPP; INC_MAX, maximum phase number.
Output: $\boldsymbol{x}^k$, a possibly better solution of SPP.

Compute $\mathcal{P}$ and $\mathcal{Z}_\iota$ associated with $\boldsymbol{x}^0$; $k := 0$; $\iota := 1$;
while $\iota < INC\_MAX$ *and* $\mathcal{Z}_{\iota-1} \neq \mathcal{Z}$ **do**
  Solve $\text{MNA}^{\boldsymbol{w}}_{\mathcal{Z}_\iota}$;
  if $\text{MNA}^{\boldsymbol{w}}_{\mathcal{Z}_\iota}$ *is infeasible* **then**
    Remove all fixing constraints from $\text{MNA}^{\boldsymbol{w}}_{\mathcal{Z}_\iota}$; $\iota := \iota + 1$; Recompute $\mathcal{Z}_\iota$;
  else
    $\boldsymbol{d}^\star :=$ an (extreme) optimal solution of $\text{MNA}^{\boldsymbol{w}}_{\mathcal{Z}_\iota}$;
    if $\boldsymbol{c}^T\boldsymbol{d}^\star < 0$ **then**
      if $\boldsymbol{d}^\star_{\mathcal{Z}_\iota}$ *is column-disjoint* **then**
        $r^\star :=$ maximal step in direction $\boldsymbol{d}^\star$; $\boldsymbol{x}^{k+1} := \boldsymbol{x}^k + r^\star\boldsymbol{d}^\star$; $k := k + 1$; Recompute $\mathcal{P}$ and $\mathcal{Z}_\iota$ according to the new solution;
      else
        For each $j \in \mathit{Supp}\,(\boldsymbol{d}^\star_{\mathcal{Z}})$, add the fixing constraint $d_j = 0$ to $\text{MNA}^{\boldsymbol{w}}_{\mathcal{Z}_\iota}$;
    else
      Remove all fixing constraints from $\text{MNA}^{\boldsymbol{w}}_{\mathcal{Z}_\iota}$; $\iota := \iota + 1$; Recompute $\mathcal{Z}_\iota$;

Remark. Rosat et al. [13] prove that the maximum normalized augmentation problem $\text{MNA}^{\boldsymbol{w}}$ can be decomposed into two subproblems that can be solved separately. Set $\mathcal{Z}$ is partitioned into two smaller subsets $\mathcal{C}$ and $\mathcal{I}$, where the indices in $\mathcal{C}$ are those of all the nonbasic variables in $\mathsf{Span}\,(\boldsymbol{A}_{.\mathcal{P}})$, i.e., the compatible variables, and $\mathcal{I}$ contains the others. Let $\text{MNA}^{\boldsymbol{w}}(\mathcal{C})$ and $\text{MNA}^{\boldsymbol{w}}(\mathcal{I})$ be the problems obtained from $\text{MNA}^{\boldsymbol{w}}$ by replacing $\mathcal{Z}$ with $\mathcal{C}$ and $\mathcal{I}$. Rosat et al. [13] prove that $z^\star_{\text{MNA}^{\boldsymbol{w}}} = \min\left\{z^\star_{\text{MNA}^{\boldsymbol{w}}(\mathcal{C})}, z^\star_{\text{MNA}^{\boldsymbol{w}}(\mathcal{I})}\right\}$. Solving $\text{MNA}^{\boldsymbol{w}}$ is therefore equivalent to solving these two problems independently. We also use this decomposition in our implementation.

For the sake of clarity, although we make use of the above two remarks, we describe all manipulations related to the augmentation using $\text{MNA}^{\boldsymbol{w}}$. We can transpose them all to the projection of $\Delta_{\boldsymbol{w}}$ by applying the same linear transformation that yields that projection, and we can adapt them to the decomposition by simply applying them to either subproblem.

4 Dynamic normalization in ISUD

This section presents the core of our algorithmic improvements to ISUD. In our new version of the algorithm, when the (extreme) optimal solution $\boldsymbol{d}^\star$ of $\text{MNA}^{\boldsymbol{w}}$ is fractional, we add a perturbation $\tilde{\boldsymbol{w}}$ to the weights to obtain a column-disjoint solution for the modified problem $\text{MNA}^{(\boldsymbol{w}+\tilde{\boldsymbol{w}})}$. In Section 4.1, we explain the design of that method and we give the pseudocode. In Section 4.2, we give geometric insight into and theoretical properties of the normalization vectors and their connection to the set of feasible directions. In Section 4.3, we detail our four update strategies, which will be compared in Section 6.

4.1 Design and algorithm

Suppose that the optimal solution $\boldsymbol{d}^\star \in \mathit{Ext}\,(\Delta_{\boldsymbol{w}})$ of $\text{MNA}^{\boldsymbol{w}}$ is a fractional direction of negative cost and that $\boldsymbol{x}^k$ is **not** optimal for SPP. It is important to recall here that all extreme integral directions of edges of the SPP linear relaxation are quasi-integral. Therefore, $\Delta_{\boldsymbol{w}}$ is a relaxation of $\Delta^{int}_{\boldsymbol{w}}$ with the property that $\mathit{Ext}\left(\Delta^{int}_{\boldsymbol{w}}\right) \subseteq \mathit{Ext}\,(\Delta_{\boldsymbol{w}})$. We aim to determine $\boldsymbol{w'}$ such that the optimal solution $\boldsymbol{d'}$ of $\text{MNA}^{\boldsymbol{w'}}$ is column-disjoint. From Rosat et al. [13], we know that such a $\boldsymbol{w'}$ exists. We propose a modification of the program ($\text{MNA}^{\boldsymbol{w}}$) that models the relaxation of **iAUG**. Another approach would be to tighten the relaxation polytope by adding cutting planes (see [14]). We chose to act on the normalization constraint only because (1) we know that it is sufficient, (2) it is faster than adding cutting planes, and (3) adding cuts tends to increase the number of fractional solutions if the cuts are not facets. A modification of the normalization constraint does not change the nature of the polytope; it simply scales the directions in a different way.

Our method results in the addition of lines 12–15 in the new version of ISUD (Algorithm 2). Regardless of the strategy chosen to compute the perturbation $\tilde{\boldsymbol{w}}$, PERT () is the function that computes this perturbation, and q is the number of consecutive updates that have already been performed. We may not need all these inputs, but in general the perturbation depends on the current solution $\boldsymbol{x}^k$, the (non-column-disjoint) current augmenting direction $\boldsymbol{d}^\star$, the weight vector $\boldsymbol{w}$, and q, so $\tilde{\boldsymbol{w}} = \texttt{PERT}\left(\boldsymbol{x}^k, \boldsymbol{d}^\star, \boldsymbol{w}, q\right)$ (line 13 of Algorithm 2). To ensure the termination of the process, we specify a maximum number Q of consecutive updates. Every time variables are fixed or the phase number is increased the counter q is reset to zero (lines 5 and 15).

Algorithm 2: ISUD with dynamic updates of the normalization weights

Input: $\boldsymbol{x}^0$, a solution of SPP; INC_MAX, maximum phase number considered; Q, maximum number of consecutive updates of normalization weights.
Output: $\boldsymbol{x}^k$, a possibly better solution of SPP.

1 Compute $\mathcal{P}$ and $\mathcal{Z}_\iota$ associated with $\boldsymbol{x}^0$; $k := 0$; $\iota := 1$; $q := 0$;
2 **while** $\iota < INC\_MAX$ *and* $\mathcal{Z}_{\iota-1} \neq \mathcal{Z}$ **do**
3 Solve $\text{MNA}^{\boldsymbol{w}}_{|\mathcal{Z}_\iota}$;
4 **if** $\text{MNA}^{\boldsymbol{w}}_{|\mathcal{Z}_\iota}$ *is infeasible* **then**
5 Remove all fixing constraints from $\text{MNA}^{\boldsymbol{w}}_{|\mathcal{Z}_\iota}$; $\iota := \iota + 1$; Recompute $\mathcal{Z}_\iota$; $q := 0$;
6 **else**
7 $\boldsymbol{d}^\star :=$ an (extreme) optimal solution of $\text{MNA}^{\boldsymbol{w}}_{|\mathcal{Z}_\iota}$;
8 **if** $\boldsymbol{c}^T\boldsymbol{d}^\star < 0$ **then**
9 **if** $\boldsymbol{d}^\star_{\mathcal{Z}_\iota}$ *is column-disjoint* **then**
10 $r^\star :=$ maximal step in direction $\boldsymbol{d}^\star$; $\boldsymbol{x}^{k+1} := \boldsymbol{x}^k + r^\star d^\star$; $k := k + 1$; Recompute $\mathcal{P}$ and $\mathcal{Z}_\iota$ according to the new solution; $q := 0$;
11 **else**
12 **if** $q < Q$ **then**
13 $\tilde{\boldsymbol{w}} := \texttt{PERT}\left(\boldsymbol{x}^k, \boldsymbol{d}^\star, \boldsymbol{w}, q\right)$; $\boldsymbol{w} := \boldsymbol{w} + \tilde{\boldsymbol{w}}$; $q := q + 1$;
14 **else**
15 For each $j \in \mathit{Supp}\left(\boldsymbol{d}^\star_{\mathcal{Z}}\right)$, add the fixing constraint $d_j = 0$ to $\text{MNA}^{\boldsymbol{w}}_{|\mathcal{Z}_\iota}$; $q := 0$;
16 **else**
17 Remove all fixing constraints from $\text{MNA}^{\boldsymbol{w}}_{|\mathcal{Z}_\iota}$; $\iota := \iota + 1$; $q := 0$; Recompute $\mathcal{Z}_\iota$;

4.2 Geometry of normalization weight vectors

We now give some geometric insights into the structure of the set of normalization constraints. This will provide a better understanding of our methods and their mechanics. We associate the weight vectors with optimal solutions of $\text{MNA}^{\boldsymbol{w}}$ and show that the set of normalization constraints that yields the same optimal solution is a polytope. Let $\mathcal{W} = \left\{\boldsymbol{w} \in \mathbb{R}^n | \boldsymbol{w}_\mathcal{P} \leq \boldsymbol{0}, \boldsymbol{w}_\mathcal{Z} > \boldsymbol{0}, -\boldsymbol{e}^T\boldsymbol{w}_\mathcal{P} + \boldsymbol{e}^T\boldsymbol{w}_\mathcal{Z} = 1\right\}$ be the set of scaled normalization weight vectors that we wish to consider. As before, we assume that $\boldsymbol{w}_\mathcal{P} \leq \boldsymbol{0}$ and $\boldsymbol{w}_\mathcal{Z} > \boldsymbol{0}$ because these are sufficent conditions for $\Delta_{\boldsymbol{w}}$ to be a proper section of the cone of feasible directions, without being too restrictive on the choice of $\boldsymbol{w}$. Let $\bar{\boldsymbol{w}} = \boldsymbol{w} / \left(-\boldsymbol{e}^T\boldsymbol{w}_\mathcal{P} + \boldsymbol{e}^T\boldsymbol{w}_\mathcal{Z}\right)$. Since $\text{MNA}^{\bar{\boldsymbol{w}}}$ is equivalent to $\text{MNA}^{\boldsymbol{w}}$, we can reasonably reduce the set of all possible normalization weights to $\mathcal{W}$. For every extreme direction $\boldsymbol{d}^{\boldsymbol{e}}$ of the cone of feasible directions, let $\mathcal{W}\left(\boldsymbol{d}^{\boldsymbol{e}}\right) = \left\{\boldsymbol{w} \in \mathcal{W} | \left(\boldsymbol{d}^{\boldsymbol{e}} / \boldsymbol{w}^T\boldsymbol{d}^{\boldsymbol{e}}\right) \in \arg\min \text{MNA}^{\boldsymbol{w}}\right\}$ be the set of normalization vectors $\boldsymbol{w} \in \mathcal{W}$ such that $\left(\boldsymbol{d}^{\boldsymbol{e}} / \boldsymbol{w}^T\boldsymbol{d}^{\boldsymbol{e}}\right)$ is an optimal solution of $\text{MNA}^{\boldsymbol{w}}$. In the same spirit, $\overset{\circ}{\mathcal{W}}\left(\boldsymbol{d}^{\boldsymbol{e}}\right) = \left\{\boldsymbol{w} \in \mathcal{W} | \arg\min \text{MNA}^{\boldsymbol{w}} \text{ is the singleton } \left\{\boldsymbol{d}^{\boldsymbol{e}} / \boldsymbol{w}^T\boldsymbol{d}^{\boldsymbol{e}}\right\}\right\}$ is the set of normalization vectors $\boldsymbol{w} \in \mathcal{W}$ such that $\left(\boldsymbol{d}^{\boldsymbol{e}} / \boldsymbol{w}^T\boldsymbol{d}^{\boldsymbol{e}}\right)$ is the **only** optimal solution of $\text{MNA}^{\boldsymbol{w}}$. Obviously, $\overset{\circ}{\mathcal{W}}\left(\boldsymbol{d}^{\boldsymbol{e}}\right) \subseteq \mathcal{W}\left(\boldsymbol{d}^{\boldsymbol{e}}\right)$. In Proposition 4, we prove that $\overset{\circ}{\mathcal{W}}\left(\boldsymbol{d}^{\boldsymbol{e}}\right)$ is the topological interior of $\mathcal{W}\left(\boldsymbol{d}^{\boldsymbol{e}}\right)$, hence the chosen notation.

Proposition 3 *Suppose that* $\boldsymbol{x}^k \in \mathcal{F}_{\text{SPP}}$ *is not optimal for* SPP^{LR}. *Given an extreme direction* $\boldsymbol{d}^{\boldsymbol{e}}$, *the following statements are equivalent:*

(i) $\boldsymbol{c}^T\boldsymbol{d}^{\boldsymbol{e}} < 0$;
(ii) $\overset{\circ}{\mathcal{W}}\left(\boldsymbol{d}^{\boldsymbol{e}}\right) \neq \emptyset$;
(iii) $\mathcal{W}\left(\boldsymbol{d}^{\boldsymbol{e}}\right) \neq \emptyset$.

Proof. We prove that (i) ⇒ (ii), (ii) ⇒ (iii), and (iii) ⇒ (i).

(i) ⇒ (ii): This is proved in Lemma 11 of [13].

(ii) ⇒ (iii): $\overset{\circ}{\mathcal{W}}(\boldsymbol{d}^{\boldsymbol{e}}) \subseteq \mathcal{W}(\boldsymbol{d}^{\boldsymbol{e}})$, so (ii) ⇒ (iii).

(iii) ⇒ (i): Suppose that $\mathcal{W}(\boldsymbol{d}^{\boldsymbol{e}}) \neq \emptyset$ and let $\boldsymbol{w} \in \mathcal{W}(\boldsymbol{d}^{\boldsymbol{e}})$. Since $\boldsymbol{x}^k$ is nonoptimal for SPP^{LR}, there exists a normalized augmenting feasible direction $\boldsymbol{d}^\star \in \Delta_{\boldsymbol{w}}$ at $\boldsymbol{x}^0$. By definition of $\mathcal{W}(\boldsymbol{d}^{\boldsymbol{e}})$, $\boldsymbol{d}^{\boldsymbol{e}}/\left(\boldsymbol{w}^T\boldsymbol{d}^{\boldsymbol{e}}\right)$ is optimal for $\text{MNA}^{\boldsymbol{w}}$, hence $\boldsymbol{c}^T\boldsymbol{d}^{\boldsymbol{e}}/\left(\boldsymbol{w}^T\boldsymbol{d}^{\boldsymbol{e}}\right) \leq \boldsymbol{c}^T\boldsymbol{d}^\star < 0$. Because $\boldsymbol{w}^T\boldsymbol{d}^{\boldsymbol{e}} > 0$, the result holds.

□

Proposition 4 *Given an extreme direction $\boldsymbol{d}^{\boldsymbol{e}}$, $\mathcal{W}(\boldsymbol{d}^{\boldsymbol{e}})$ is a polytope and $\overset{\circ}{\mathcal{W}}(\boldsymbol{d}^{\boldsymbol{e}})$ is its topological interior within $\mathcal{W}$.*

Proof. Let $\hat{\boldsymbol{w}} \in \mathcal{W}$ be a reference normalization vector. Since extreme directions of the cone of feasible directions correspond to extreme points of any of its sections, we have

$$\begin{aligned}
\mathcal{W}(\boldsymbol{d}^{\boldsymbol{e}}) &= \left\{\boldsymbol{w} \in \mathcal{W} | \frac{\boldsymbol{d}^{\boldsymbol{e}}}{\boldsymbol{w}^T\boldsymbol{d}^{\boldsymbol{e}}} \in \arg\min \text{MNA}^{\boldsymbol{w}}\right\} \\
&= \left\{\boldsymbol{w} \in \mathcal{W} | \forall \boldsymbol{u} \in Ext\left(\Delta_{\hat{\boldsymbol{w}}}\right) \setminus \left\{\boldsymbol{d}^{\boldsymbol{e}}/\hat{\boldsymbol{w}}^T\boldsymbol{d}^{\boldsymbol{e}}\right\}, \frac{\boldsymbol{c}^T\boldsymbol{d}^{\boldsymbol{e}}}{\boldsymbol{w}^T\boldsymbol{d}^{\boldsymbol{e}}} \leq \frac{\boldsymbol{c}^T\boldsymbol{u}}{\boldsymbol{w}^T\boldsymbol{u}}\right\} \\
&= \left\{\boldsymbol{w} \in \mathcal{W} | \forall \boldsymbol{u} \in Ext\left(\Delta_{\hat{\boldsymbol{w}}}\right) \setminus \left\{\boldsymbol{d}^{\boldsymbol{e}}/\hat{\boldsymbol{w}}^T\boldsymbol{d}^{\boldsymbol{e}}\right\}, \left(\left(\boldsymbol{c}^T\boldsymbol{d}^{\boldsymbol{e}}\right)\boldsymbol{u} - \left(\boldsymbol{c}^T\boldsymbol{u}\right)\boldsymbol{d}^{\boldsymbol{e}}\right)^T\boldsymbol{w} \leq 0\right\}.
\end{aligned}$$

Therefore, $\mathcal{W}(\boldsymbol{d}^{\boldsymbol{e}})$ is the intersection of $\mathcal{W}$ with a set P of semi-spaces (linear inequalities), which is a polyhedron. Because $\mathcal{W}(\boldsymbol{d}^{\boldsymbol{e}}) \subseteq \mathcal{W}$ and $\mathcal{W}$ is a polytope, $\mathcal{W}(\boldsymbol{d}^{\boldsymbol{e}})$ is a polytope. The same reasoning can be applied to show that

$$\overset{\circ}{\mathcal{W}}(\boldsymbol{d}^{\boldsymbol{e}}) = \left\{\boldsymbol{w} \in \mathcal{W} | \forall \boldsymbol{u} \in Ext\left(\Delta_{\hat{\boldsymbol{w}}}\right) \setminus \left\{\boldsymbol{d}^{\boldsymbol{e}}/\hat{\boldsymbol{w}}^T\boldsymbol{d}^{\boldsymbol{e}}\right\}, \left(\left(\boldsymbol{c}^T\boldsymbol{d}^{\boldsymbol{e}}\right)\boldsymbol{u} - \left(\boldsymbol{c}^T\boldsymbol{u}\right)\boldsymbol{d}^{\boldsymbol{e}}\right)^T\boldsymbol{w} < 0\right\}.$$

With that description, we see that $\overset{\circ}{\mathcal{W}}(\boldsymbol{d}^{\boldsymbol{e}})$ is the intersection of $\mathcal{W}$ with the interior of the same semi-spaces that define $\mathcal{W}(\boldsymbol{d}^{\boldsymbol{e}})$. Therefore, $\overset{\circ}{\mathcal{W}}(\boldsymbol{d}^{\boldsymbol{e}})$ is the topological interior of $\mathcal{W}(\boldsymbol{d}^{\boldsymbol{e}})$ within $\mathcal{W}$. □

Proposition 4 yields the geometric interpretation given in Figure 2. Our goal is to find $\boldsymbol{w}$ within one of the polytopes $\mathcal{W}(\boldsymbol{d})$ such that $\boldsymbol{d}$ is an integral direction. Experience tells us that in SPPs, the direction is likely to be integral. When it is not, we propose to modify $\boldsymbol{w}$ by adding an update $\tilde{\boldsymbol{w}}$ as shown in Figure 2 so as to find a point in another of the subpolytopes. Reaching a polytope whose interior is nonempty is in principle easier than converging toward a single point; the process is less sensitive to approximation errors. Two of our four strategies build the constraint anew and are akin to sampling methods. The others make smaller changes and aim to reach an adjacent polytope that is likely to be that of an integral direction. The idea behind these two methods is that the more the normalization weights are changed, the more we risk deteriorating the mean-scaled cost of the direction found.

4.3 Update strategies

We now summarize our four strategies for updating $\boldsymbol{w}$. The first attempt is always made with the incompatibility degrees as the normalization weights (DEG, $w_j = \iota_j$ if $j \in \mathcal{Z}$, 0 otherwise) because the four static versions of the algorithm compared in [13] (listed in Section 3.3) perform well with this approach.

4.3.1 Random update (Rand)

The first modification is the simplest: random perturbation. For every index $j \in \mathcal{Z}$, the penalty $\tilde{w}_j$ is the value of a uniform integer random variable between 0 and θ, where θ is a parameter. All random variables are independent. This strategy is based on empirical observations that suggest that the likelihood of an integral optimal direction is relatively high regardless of the constraint. It also serves as a base for the comparison of the other methods. It can be interpreted as random sampling of the points in $\mathcal{W}$ at each iteration.

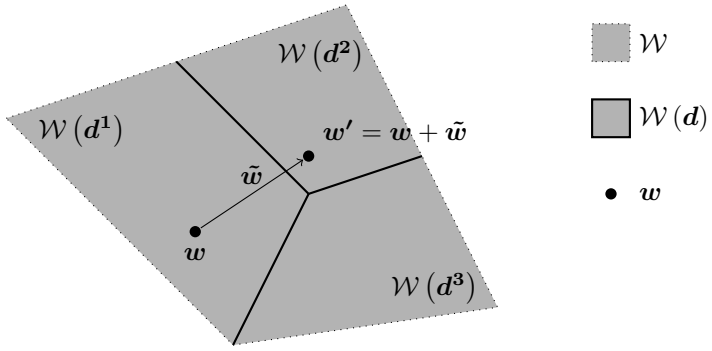

Figure 2: Geometric description of $\mathcal{W}$. The set of all available normalization constraints is the grey domain. Subpolytopes that correspond to $\mathcal{W}\left(\boldsymbol{d^i}\right)$ are delimited by thick black lines ($i \in \{1, 2, 3\}$). In this example, the only optimal solution of MNA$^{\boldsymbol{w}}$ is $\boldsymbol{d^1}$. When $\boldsymbol{w}$ is perturbed, $\boldsymbol{w'} := \boldsymbol{w} + \tilde{\boldsymbol{w}}$, the optimal solution changes because $\boldsymbol{w'} \notin \mathcal{W}\left(\boldsymbol{d^1}\right)$, and the only optimal solution of MNA$^{\boldsymbol{w'}}$ is $\boldsymbol{d^2}$.

4.3.2 Alternate update (Alt)

This update applies each of the four constraints listed in Section 3.3. We consider them in the order of their respective performance in the static version of the algorithm (as reported in [13]): DEG, NORM, MIMA, and MMA. The maximum number of consecutive updates Q is clearly four. This approach can be interpreted as the sampling of the same four points in $\mathcal{W}$ at each iteration.

4.3.3 Penalization of current fractional direction (Dir)

In the DIR update strategy, we perturb $\boldsymbol{w}$ so as to directly penalize the variables involved in the fractional direction $\boldsymbol{d^\star}$ returned by MNA$^{\boldsymbol{w}}$. For every index j in the support $\mathcal{S}^+$ of $\boldsymbol{d^\star}_{\mathcal{Z}}$, $\tilde{w}_j$ is increased by $\theta > 0$. Hence, $\tilde{\boldsymbol{w}} = \theta \mathbb{1}_{\mathcal{S}^+}$. Because the changeover from $\boldsymbol{w}$ to $\boldsymbol{w'} := \boldsymbol{w} + \tilde{\boldsymbol{w}}$ does not guarantee that the best augmenting direction $\boldsymbol{d^\star}$ changes, the value of θ is updated during the process depending on the success of previous attempts. On the one hand, the higher θ, the higher the probability that $\boldsymbol{d^\star}$ (or its scaling) is no longer optimal for MNA$^{\boldsymbol{w'}}$. On the other hand, the lower θ, the closer the solution of MNA$^{\boldsymbol{w'}}$ to $\boldsymbol{d^\star}$ and the less deterioration in terms of the optimal value and quality of the augmentation. We handle this tradeoff in the following way: θ is given an initial value θ_0 (we report results for different values of θ_0). When we update the normalization weights, the new value of θ is

$$\theta := \begin{cases} 2\theta & \text{if the solution did not change } (\boldsymbol{d^\star} \text{ and } \boldsymbol{d'} \text{ are proportional}), \\ \theta/2 & \text{if the new solution } \boldsymbol{d'} \text{ is an integral direction}, \\ \theta/\lambda & \text{if } \boldsymbol{d'} \neq \boldsymbol{d^\star} \text{ and the mean scaled cost was deteriorated by a factor greater than 2}, \\ \theta & \text{otherwise}, \end{cases} \tag{4}$$

where $\boldsymbol{d^\star}$ and $\boldsymbol{d'}$ are optimal solutions of MNA$^{\boldsymbol{w}}$ and MNA$^{\boldsymbol{w'}}$ respectively, and λ is a uniform random variable between 1 and 2. Three of these updates of θ are obviously consistent with the previous remarks, but the θ/λ update needs further explanation. When the solution changes but the cost is noticeably worse, the penalty is high enough to force the algorithm to seek out what can be seen as "bad" solutions; θ should therefore decrease. However, it is not desirable to restrict the possible values taken by θ to a discrete set. Our goal is instead to iteratively tune θ so that it reaches a tradeoff value, neither too large nor too small.

Remark. The penalization that we apply is the same for all variables of $\mathcal{S}^+$; it is not, for example, proportional to $\boldsymbol{d^\star}$. This is because we wish to foster column-disjoint solutions. A larger value for $d^\star_{j_1}$ than $d^\star_{j_2}$ in the fractional direction $\boldsymbol{d^\star}$ does not indicate that j_2 is more likely than j_1 to be part of a disjoint solution. Arguments can be found to justify both this position and the opposite. The propensity of a variable for appearing in disjoint or nondisjoint combinations is instead related to the nonzero entries of the corresponding column $\boldsymbol{A}_{\cdot j}$ and is taken into account because the base constraint is DEG. For this reason, we equally penalize each variable appearing in the nondisjoint direction $\boldsymbol{d^\star}$.

4.3.4 Update based on cutting planes (Hyp)

In the Hyp update strategy, we perturb $\boldsymbol{w}$ using the coefficient of the cutting planes that separate $\boldsymbol{d}^\star$ from the set of feasible integer directions $\Delta_{\boldsymbol{w}}^{int}$. We again assume that the optimal solution $\boldsymbol{d}^\star \in Ext\left(\Delta_{\boldsymbol{w}}\right)$ of $\text{Mna}^{\boldsymbol{w}}$ found is not column-disjoint and of negative cost; we also assume that $\boldsymbol{x}^k$ is **not** optimal for SPP. Rosat et al. [14] provide separation algorithms for primal versions of odd-cycle and clique cuts that can be used in ISUD to separate such a direction from $\Delta_{\boldsymbol{w}}^{int}$. They prove that these cuts are of the form $\boldsymbol{\alpha}^T\boldsymbol{d} \leq 0$. We assume that a nonempty set $\mathcal{K}$ ($\mathcal{K}$ denotes both the set of cuts and that of their coefficient vectors since the context is clear enough) of such cutting planes has been generated, i.e., for every $\boldsymbol{\alpha} \in \mathcal{K}$, the corresponding inequality $\boldsymbol{\alpha}^T\boldsymbol{d} \leq 0$ is valid for $\Delta_{\boldsymbol{w}}^{int}$ and violated by $\boldsymbol{d}^\star$ ($\boldsymbol{\alpha}^T\boldsymbol{d}^\star > 0$). From this set of cutting planes, we compute the modification of the normalization weights as

$$\tilde{\boldsymbol{w}} = \frac{\theta}{|\mathcal{K}|} \sum_{\boldsymbol{\alpha} \in \mathcal{K}} \boldsymbol{\alpha}. \tag{5}$$

The perturbation is scaled according to the number of cutting planes added, and it is multiplied by a coefficient θ, the value of which is updated as for Dir (Section 4.3.3). The theoretical framework for this strategy is given in Proposition 5, and a geometric interpretation is given in Figure 3.

Proposition 5 *Assume that the current solution $\boldsymbol{x}^k$ is not optimal for* SPP. *Let $\boldsymbol{d}^\star \in Ext\left(\Delta_{\boldsymbol{w}}\right)$ be a non-column-disjoint optimal solution of* $\text{Mna}^{\boldsymbol{w}}$ *and $\boldsymbol{d}^{\boldsymbol{0}} \in \Delta_{\boldsymbol{w}}^{int}$ an integral augmenting direction. Let $\boldsymbol{\alpha} \in \mathbb{R}^n$ be the coefficient vector of a cut $\boldsymbol{\alpha}^T\boldsymbol{x} \leq 0$ that separates $\boldsymbol{d}^\star$ from $\Delta_{\boldsymbol{w}}^{int}$, and $\theta > 0$. Consider the perturbation of the normalization weights $\tilde{\boldsymbol{w}} = \theta\boldsymbol{\alpha}$, and let the updated vector be $\boldsymbol{w'} := \boldsymbol{w} + \tilde{\boldsymbol{w}}$. Let $\boldsymbol{d}^{\star\prime} = \boldsymbol{d}^\star / \left(\boldsymbol{w'}^T\boldsymbol{d}^\star\right)$ and $\boldsymbol{d}^{\boldsymbol{0}\prime} = \boldsymbol{d}^{\boldsymbol{0}} / \left(\boldsymbol{w'}^T\boldsymbol{d}^{\boldsymbol{0}}\right)$ be the scalings of $\boldsymbol{d}^{\star\prime}$ and $\boldsymbol{d}^{\boldsymbol{0}\prime}$ for the new section of the cone of feasible directions. Then,*

(i) the cost of the direction associated with $\boldsymbol{d}^\star$ worsens after the update of the normalization weights, i.e.,

$$\boldsymbol{c}^T\boldsymbol{d}^{\star\prime} > \boldsymbol{c}^T\boldsymbol{d}^\star; \tag{6}$$

(ii) the cost of the direction associated with $\boldsymbol{d}^{\boldsymbol{0}}$ improves after the update of the normalization weights, i.e.,

$$\boldsymbol{c}^T\boldsymbol{d}^{\boldsymbol{0}\prime} \leq \boldsymbol{c}^T\boldsymbol{d}^{\boldsymbol{0}}. \tag{7}$$

Proof. Let $\boldsymbol{d}^\star$, $\boldsymbol{d}^{\boldsymbol{0}}$, $\boldsymbol{\alpha}$, θ, $\tilde{\boldsymbol{w}}$, $\boldsymbol{w'}$, $\boldsymbol{d}^{\star\prime}$, and $\boldsymbol{d}^{\boldsymbol{0}\prime}$ be as defined in the statement of Proposition 5.

(i) The cost of $\boldsymbol{d}^{\star\prime}$ satisfies

$$\boldsymbol{c}^T\boldsymbol{d}^{\star\prime} = \boldsymbol{c}^T\left(\frac{\boldsymbol{d}^\star}{\boldsymbol{w'}^T\boldsymbol{d}^\star}\right) = \frac{\boldsymbol{c}^T\boldsymbol{d}^\star}{\boldsymbol{w}^T\boldsymbol{d}^\star + \tilde{\boldsymbol{w}}^T\boldsymbol{d}^\star} = \frac{\boldsymbol{c}^T\boldsymbol{d}^\star}{1 + \theta\left(\boldsymbol{\alpha}^T\boldsymbol{d}^\star\right)}. \tag{8}$$

By definition, $\alpha^T\boldsymbol{d}^\star > 0$ and $\theta > 0$, so $1+\theta\left(\boldsymbol{\alpha}^T\boldsymbol{d}^\star\right) > 1$. Because $\boldsymbol{x}^k$ is not optimal for SPP, $z^\star_{\text{Mna}^{\boldsymbol{w}}} < 0$ and therefore $\boldsymbol{c}^T\boldsymbol{d}^\star < 0$. Thus, Equation 6 holds.

(ii) We will apply the same reasoning to $\boldsymbol{d}^{\boldsymbol{0}}$:

$$\boldsymbol{c}^T\boldsymbol{d}^{\boldsymbol{0}\prime} = \boldsymbol{c}^T\left(\frac{\boldsymbol{d}^{\boldsymbol{0}}}{\boldsymbol{w'}^T\boldsymbol{d}^{\boldsymbol{0}}}\right) = \frac{\boldsymbol{c}^T\boldsymbol{d}^{\boldsymbol{0}}}{\boldsymbol{w}^T\boldsymbol{d}^{\boldsymbol{0}} + \tilde{\boldsymbol{w}}^T\boldsymbol{d}^{\boldsymbol{0}}} = \frac{\boldsymbol{c}^T\boldsymbol{d}^{\boldsymbol{0}}}{1 + \theta\left(\boldsymbol{\alpha}^T\boldsymbol{d}^{\boldsymbol{0}}\right)}. \tag{9}$$

By definition, $\boldsymbol{\alpha}^T\boldsymbol{d}^{\boldsymbol{0}} \leq 0$ and $\theta > 0$, so $1 + \theta\left(\boldsymbol{\alpha}^T\boldsymbol{d}^\star\right) \leq 1$. Because $\boldsymbol{d}^{\boldsymbol{0}}$ is an augmenting direction, $\boldsymbol{c}^T\boldsymbol{d}^{\boldsymbol{0}} < 0$. Thus, Equation 7 holds.

□

Recall that when we apply the Hyp update, the cuts computed to generate the perturbation $\tilde{\boldsymbol{w}}$ are not added to the formulation of $\text{Mna}^{\boldsymbol{w}}$ even if they are technically available. The addition of cutting planes

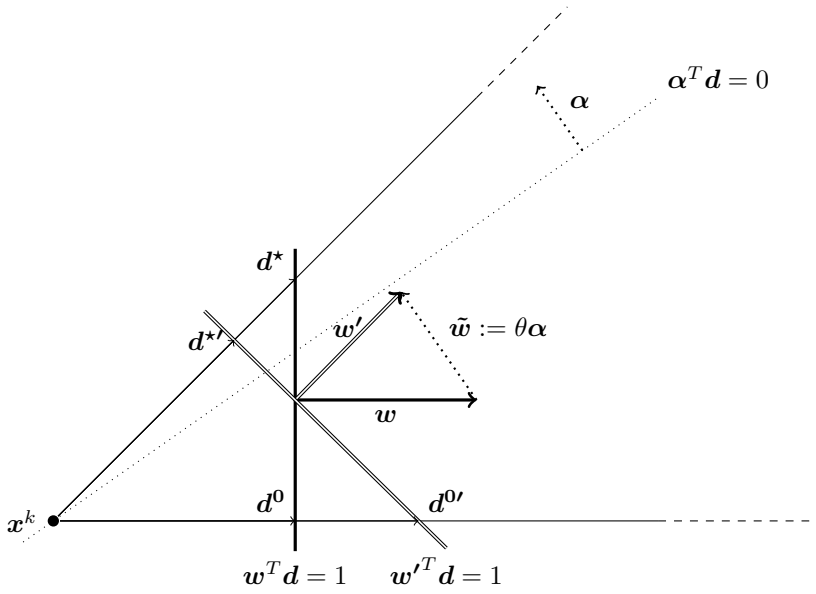

Figure 3: Geometric description of HYP. The two plain rays starting from $\boldsymbol{x}^0$ define the cone of feasible directions. Vector $\boldsymbol{w}$ is the original normalization weight vector, and the cone section defined by $\boldsymbol{w}^T\boldsymbol{d} = 1$ is the plain line, orthogonal to $\boldsymbol{w}$. Vectors $\boldsymbol{d}^\star$ and $\boldsymbol{d}^{\mathbf{0}}$ are the normalized rays for vector $\boldsymbol{w}$. Here, we find a cutting plane $\boldsymbol{\alpha}^T\boldsymbol{d} \leq 0$ that separates $\boldsymbol{d}^\star$ from $\Delta_{\boldsymbol{w}}^{int}$ (dotted ray, and corresponding $\boldsymbol{\alpha}$). A penalty $\tilde{\boldsymbol{w}}$ proportional to α ($\theta > 0$) is added to $\boldsymbol{w}$ to define $\boldsymbol{w}'$ (double arrow). The updated normalization constraint $\boldsymbol{w}'^T\boldsymbol{d} = 1$ is the double line, and the associated extreme directions are $\boldsymbol{d}^{\star\prime}$ and $\boldsymbol{d}^{\mathbf{0}\prime}$. In this example, we see that the addition of the perturbation $\tilde{\boldsymbol{w}}$ tends to scale down the cut-off solution ($\boldsymbol{d}^\star \to \boldsymbol{d}^{\star\prime}$) and scale up the other solution ($\boldsymbol{d}^{\mathbf{0}} \to \boldsymbol{d}^{\mathbf{0}\prime}$).

slows the solution process and often pushes $\boldsymbol{d}^\star$ toward a fractional direction that is an extreme point created by the cutting planes. HYP tends to push the optimal solution further, until it reaches an extreme point of the original cone that is more likely to be an integral direction.

Furthermore, this method raises the issue that $\boldsymbol{w}'$ may not guarantee that $\Delta_{\boldsymbol{w}'}$ is a proper section of the cone. As stated in [13], a sufficient condition to guarantee this is $\boldsymbol{w}_{\mathcal{P}} \leq \mathbf{0}$ and $\boldsymbol{d}_{\mathcal{Z}} > \mathbf{0}$. In all cases, we use DEG for the first attempt, i.e., $\boldsymbol{w} = (\mathbf{0}, \boldsymbol{\iota}_{\mathcal{Z}})$. In both primal odd-cycle and clique inequalities, the coefficients of some variables in $\mathcal{P}$ take positive values. Therefore, if $\theta \neq 0$, one of these coefficients w_j ($j \in \mathcal{P}$) strictly increases. Since its original value is 0, it becomes positive and the condition $\boldsymbol{w}_{\mathcal{P}} \leq \mathbf{0}$ no longer holds.

This theoretically forbids the increase of every w_j for $j \in \mathcal{P}$ under these sufficient conditions. However, $\tilde{\boldsymbol{w}}$ is a continuous function of θ, and there therefore exists a neighborhood of $\theta = 0$ where $(\boldsymbol{w} + \tilde{\boldsymbol{w}})^T\boldsymbol{d} = 1$ is a bounded section of the cone. Although nothing guarantees that the value chosen for θ belongs to that neighborhood, the case of a direction going to infinity did not occur in any of our numerical experiments. If it were to occur, we would decrease the value of θ until $\Delta_{\boldsymbol{w}}$ is bounded or the maximum number Q of consecutive updates of the normalization constraint is reached.

5 Benchmark

The crew pairing problem involves determining a set of pairings (where a pairing is a sequence of flights and layovers that starts and ends at the same base) that covers all the scheduled flights at minimal cost over the planning horizon. For a review of aircrew scheduling, see Kasirzadeh et al. [8]. We concentrate on the crew pairing problem.

5.1 Set partitioning and crew pairing problem

Model. The crew pairing problem is usually solved by column generation embedded within a branch-and-bound scheme (see Desaulniers et al. [3]). In the underlying Dantzig–Wolfe decomposition, most constraints of the master problem are set-partitioning constraints. Each of the rows $i \in \{1, \ldots, m\}$ represents a scheduled flight, and each column $j \in \{1, \ldots, n\}$ corresponds to a legal pairing. The corresponding matrix entry A_{ij} is 1 if flight i is covered by pairing j, and 0 otherwise. A set of pairings covering each scheduled flight exactly once is therefore a binary solution of $\boldsymbol{Ax} = \boldsymbol{e}$, i.e., of SPP. The Dantzig–Wolfe subproblems generate legal pairings and can be modeled as shortest path problems with resource constraints in time-space networks [3]. At present, we do not embed the generation of new pairings within ISUD to make it an all-integer column-generation algorithm; this is our long-term goal. For now, we consider set-partitioning instances generated from aircrew scheduling problems by the procedure described below.

Instances ACS**.** The data for the crew pairing problem is taken from the operations of a major North American airline. It consists of five instances (ACS1–5), each describing the flight schedule of a specific type of aircraft. Each instance has three crew bases. A partial schedule for the week prior to the start of the planning horizon is also available. It is composed of partial pairings that start before the beginning of the horizon and must continue during the horizon. We add an extra row to the model for each of these partial pairings, and they are considered equivalent to scheduled flights in the master problem, i.e., each of them must be covered. In the instances presented here, the number of partial pairings never exceeds 3% of the number of flights.

Pairing generation. The pairing problem involves finding a set of pairings that covers all the flights at minimal cost. The solutions must be consistent with the previous week's schedules. Each of the instances is solved using GENCOL,[1] which implements column generation embedded in a branch-and-bound framework. To generate legal pairings, we define a subproblem for each base and each day of the planning horizon (more information on the exact structure of the subproblems can be found in [15]). Each column covers one or several flights and may contain at most one partial pairing from the previous week's schedule. We obtain an integer solution by using a variable fixing procedure that selects pairings that must appear in the solution. At each iteration, we fix variables based on their fractional values in the optimal solution of the linear relaxation, and we then perform reoptimization using column generation. The procedure stops when all the variables are integer. All columns generated throughout the variable fixing procedure are recorded, and their union forms the columns of matrix $\boldsymbol{A}$. In the case of duplicate entries, only one occurrence of the column is kept in the matrix. The instances that we use to generate the pairings are those of Kasirzadeh et al. [8] from which a horizon of nine days is extracted.

Previous instances. To the ACS instances, we add two aircrew scheduling set-partitioning instances from the OR-Library,[2] SPPAA01 and SPPAA04. These two instances were included in the test sets used to evaluate former versions of ISUD (see [13, 14, 24]).

Characteristics of the instances (Table 1). The main characteristics of the instances are given in Table 1. The columns m and n respectively give the number of rows and columns of matrix $\boldsymbol{A}$. The number of rows is the number of scheduled flights plus the number of partial pairings. In the benchmark, m ranges from 423 to 1,706 and n from 7,195 to 115,940. The GENCOL UB is the best solution value found by the GENCOL branch-and-bound + column-generation procedure. The GENCOL+ISUD UB is the best solution value found by running the default version of ISUD (without the normalization update) using the best solution found by GENCOL as the initial solution. Under the CPLEX heading we give the best upper bound (UB), the best lower bound (LB), the gap between UB and LB (GAP), and the total execution time (TIME) for the instance composed of the columns generated by GENCOL with the default settings of IBM CPLEX.[3] The best UBs found are highlighted in bold.

[1] GENCOL is commercial software developed at the GERAD research center and now owned by AD OPT, a division of KRONOS.

[2] The OR-Library is a collection of test data sets for a variety of operations research problems. It is maintained by John E. Beasley and available at http://people.brunel.ac.uk/~mastjjb/jeb/info.html.

[3] Version 12.6.0 of CPLEX was run with the default parameters and a time limit of 1,800 s on a Linux PC with eight processors of 3.4 GHz.

Table 1: Sizes and solutions of the instances.

			GENCOL	GENCOL +ISUD	CPLEX			
Instance	m	n	UB	UB	UB	LB	GAP	TIME (s)
ACS1	454	12,091	80,034	80,033	**80,010**	80,010.0	0.00%	1.1
ACS2	550	20,269	99,506	**99,450**	**99,450**	99,440.1	0.02%	21.7
ACS3	1,624	103,597	227,116	**226,791**	229,800	225,652.1	1.81%	1,800.0
ACS4	1,706	83,114	336,911	**336,891**	341,530	336,859.6	1.37%	1,800.0
ACS5	1,703	115,940	305,530	**305,422**	309,910	304,326.2	1.80%	1,800.0
SPPAA01	423	7,195	.	.	**56,137**	56,132.8	0.01%	23.8
SPPAA04	803	8,904	.	.	**26,374**	26,374.0	0.00%	13.8

5.2 Initial solutions, primal information, and full benchmark

To test an integral simplex algorithm, we also need an initial solution.

Primal Information. We define the *primal information* of a solution to be the percentage of consecutive flights 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 aircrew scheduling problems. First, crews do not often change planes during their rotations, and the pairings therefore have many consecutive tasks in common with those of the aircraft 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 (airlines generally add penalties in the objective function to discourage changes). Thus, many consecutive tasks from the initial paths (aircraft routes or the original schedule) remain consecutive in the optimal schedule. In aircrew scheduling, the figure is generally around 75% for the advance planning problem [24] and significantly higher for reoptimization.

Initial Solutions. For the preceding reasons, we perturb the best-known solutions to generate initial solutions that contain a similar level of primal information to that arising in practice when aircraft routes are used as initial solutions. These solutions are generally infeasible, so we assign the perturbed columns a high cost, as is done in practice for schedules that must follow aircraft routes. We use the perturbation method of Zaghrouti et al. [24], based on crossovers, unions, and splittings of columns. The input parameter π of that method is the proportion of columns of the optimal solution that appear in the generated initial solution. For all the instances, we created initial solutions with π ranging from 0.1 to 0.8 and retained those for which the primal information (given in Table 2) was consistent with the typical values. For each value of π retained and each instance, we generated a set of 10 such initial solutions. We retained three levels of perturbation for each instance: *hard*, *moderate*, and *easy* (ordered by increasing primal information). For example, for SPPAA01, the terms hard, moderate, and easy respectively refer to the initial solutions generated for $\pi = 0.1$, $\pi = 0.15$, and $\pi = 0.2$; hence they correspond to 70.4%, 74.4%, and 78.7% of primal information. The values are indicated in Table 2, but in Section 6 we use the terms hard, moderate, and easy instead of the numerical values. Note that when we reoptimize existing schedules, the primal information is significantly higher because we can use the original schedule as an initial solution, and modifications are usually penalized. As part of a reoptimization process, all the instances presented here would be hard.

6 Numerical results

Our numerical results indicate the influence of the type of update (NONE, ALT, RAND, DIR, or HYP), the penalization amplitude θ, and the maximum number of consecutive normalization updates Q. The tests were performed on a Linux PC with eight processors of 3.4 GHz.

Table 2: Percentage of primal information in the initial solutions of the benchmark. For each instance and each value of π, the given figure is the arithmetic mean of the primal information of the 10 initial solutions generated. No value is given when the figures are inconsistent with realistic values.

	Input parameter π									
	0.1	0.15	0.2	0.25	0.35	0.4	0.45	0.5	0.55	0.6
ACS1	.	.	.	.	.	.	72.7%	74.5%	77.3%	.
ACS2	.	.	.	.	.	71.4%	73.0%	75.9%	.	.
ACS3	.	.	.	.	.	.	72.1%	74.8%	80.9%	.
ACS4	.	.	.	.	.	.	.	71.9%	76.3%	78.4%
ACS5	.	.	.	.	.	.	70.1%	75.6%	79.1%	.
SPPAA01	70.4%	74.4%	78.7%	.	.	.	.	.	.	
SPPAA04	.	.	69.7%	73.6%	78.0%	.	.	.	.	

6.1 Performance of the algorithm

Table 3 gives the general results. For each of the instances (INST), we report results for all the strategies (STRAT) for a single set of parameters θ and Q. The values of θ and Q are discussed in the next section. For each level of difficulty (easy, moderate, or hard), the first three columns give the quality of the solution, showing how many instances were solved with a gap between 0 and 1% ($\leq 1\%$), between 1 and 2% ($\leq 2\%$), or $> 2\%$. These thresholds come from industrial observations: a solution within 1% of the best lower bound is considered excellent, and one within 2% is acceptable. We will therefore say that an instance is *unsolved* if the final gap is over 2%. Note that the artificial columns that served to generate the initial solutions were always successfully eliminated. The mean running time is given in column t_{ISUD}. In the last part of Table 3, we aggregate the results over the instances for each level of difficulty. The best strategy appears to be HYP because it leaves only 29/210 instances unsolved, whereas the figure is 65 for the version of the algorithm without updates of the normalization constraint. RAND is in second place but has inconsistent performance: it leaves 7/10 easy instances unsolved for instance ACS5, whereas the figure is 6 for ISUD without updates and 1 for HYP. The version of the algorithm without updates, NONE, is obviously faster than the versions with updates. Interestingly, HYP is one of the two fastest strategies with ALT; however, in ALT, only 3 updates are performed (i.e., MNA$^{\boldsymbol{w}}$ is solved at most 4 times before variables are fixed, whereas it is solved at most 8 times in the case of HYP). Finally, the performance of our method is comparable to that of CPLEX, and it tends to be better on large instances (smaller gap and running time).

Table 4 presents the nature of the directions found by the various strategies. The figures are aggregated over all the instances available, and the parameters are the same as those in Table 3. Column K gives the aggregated number of augmentations performed, and column #MNA gives the total number of augmentation problems MNA solved throughout all the executions. We see that updating the normalization constraint significantly increases the number of augmentation problems solved, with a much smaller increase in the number of augmentations. This can be explained as follows: when the augmentation problem fails to return an integral solution, the static version performs fixing. In the version with updates, the normalization constraint may be updated 7 times before we fix the same set of variables. Therefore, for the same sequence of solutions, the versions with updates may solve up to 8 times more MNA problems than the static version. Furthermore, they perform more augmentation steps, and so yield better solutions.

Columns #FRAC and #INT give the proportion of fractional and integral directions. Obviously, the directions tend to be less fractional for NONE because the algorithm stops more quickly when it gets into difficulty. The strategies for which 8 augmentation problems may be solved before fixing occurs have similar ratios, while the ALT strategy (only up to 4 augmentation problems solved before fixing occurs) is in between. Thus, updating the normalization weights is not likely to immediately produce integral directions, but success may be expected after a few modifications. Finally, the columns #ORIG, #FIX, and #UPD give the proportion of integral directions found without variable fixing or updates, after variable fixing, and after an update. There are no major differences between the five strategies in terms of the augmentations obtained without fixing/updating. However, when updates are performed, they are usually the decisive actions in the search for integral directions, i.e., integrality usually comes from these modifications. Augmentations tend to be

Table 3: Solutions for different update strategies for instances SPPAA01, SPPAA04, ACS1, ACS2, ACS3, ACS4, ACS5.

				Easy				Moderate				Hard			
Inst	Strat	θ	Q	$\leq 1\%$	$\leq 2\%$	$> 2\%$	t_{ISUD}	$\leq 1\%$	$\leq 2\%$	$> 2\%$	t_{ISUD}	$\leq 1\%$	$\leq 2\%$	$> 2\%$	t_{ISUD}
	None	.	.	10	0	0	13.59	10	0	0	15.45	9	0	1	16.76
	Alt	.	3	10	0	0	15.33	10	0	0	17.59	9	0	1	24.16
sppaa01	Rand	10	7	10	0	0	27.64	10	0	0	27.73	10	0	0	42.41
	Dir	1	7	10	0	0	20.83	10	0	0	22.77	10	0	0	28.10
	Hyp	0.5	7	10	0	0	15.61	10	0	0	18.40	10	0	0	21.57
	None	.	.	10	0	0	4.98	5	0	5	6.29	6	1	3	7.03
	Alt	.	3	10	0	0	5.63	9	0	1	8.34	6	1	3	9.46
sppaa04	Rand	10	7	10	0	0	8.39	7	0	3	18.30	7	1	2	17.09
	Dir	1	7	10	0	0	6.29	8	0	2	12.09	8	0	2	11.29
	Hyp	0.5	7	10	0	0	6.36	7	1	2	9.75	7	0	3	11.55
	None	.	.	8	2	0	22.57	10	0	0	22.44	9	0	1	25.53
	Alt	.	3	10	0	0	23.95	10	0	0	22.88	9	0	1	53.27
acs1	Rand	10	7	10	0	0	26.78	10	0	0	23.24	10	0	0	36.14
	Dir	1	7	8	0	2	31.78	10	0	0	21.66	10	0	0	29.85
	Hyp	0.5	7	10	0	0	26.06	10	0	0	24.47	10	0	0	29.76
	None	.	.	10	0	0	35.22	8	0	2	43.95	7	0	3	53.44
	Alt	.	3	10	0	0	37.94	10	0	0	47.85	10	0	0	57.27
acs2	Rand	10	7	9	1	0	47.06	10	0	0	52.43	10	0	0	89.24
	Dir	1	7	10	0	0	42.63	10	0	0	49.80	10	0	0	66.95
	Hyp	0.5	7	10	0	0	38.63	9	1	0	48.46	10	0	0	52.90
	None	.	.	1	3	6	906.83	1	5	4	885.11	0	3	7	1236.32
	Alt	.	3	2	5	3	1167.84	1	6	3	1198.52	0	2	8	1607.18
acs3	Rand	10	7	6	4	0	1563.13	5	2	3	1608.92	1	4	5	1727.99
	Dir	1	7	2	4	4	1380.35	3	6	1	1238.15	1	2	7	1659.56
	Hyp	0.5	7	5	4	1	1196.71	4	4	2	1193.12	1	4	5	1380.25
	None	.	.	6	2	2	756.75	4	3	3	874.88	4	2	4	992.49
	Alt	.	3	7	3	0	1023.02	5	5	0	1100.34	4	3	3	1331.61
acs4	Rand	10	7	9	1	0	1402.22	8	2	0	1528.25	5	2	3	1691.95
	Dir	1	7	6	3	1	1138.15	6	4	0	1356.47	3	4	3	1592.75
	Hyp	0.5	7	8	2	0	967.14	7	2	1	1135.09	4	4	2	1293.43
	None	.	.	2	2	6	1027.76	0	1	9	1109.81	1	0	9	1431.47
	Alt	.	3	6	3	1	1233.70	0	1	9	1522.62	0	0	10	1731.57
acs5	Rand	10	7	1	2	7	1707.39	3	2	5	1774.32	1	0	9	1776.19
	Dir	1	7	2	4	4	1544.01	0	4	6	1723.64	0	2	8	1799.18
	Hyp	0.5	7	8	1	1	1289.59	2	4	4	1588.29	0	2	8	1639.56
	None	.	.	47	9	14	.	38	9	23	.	36	6	28	.
	Alt	.	3	55	11	4	.	45	12	13	.	38	6	26	.
Total	Rand	10	7	55	8	7	.	53	6	11	.	44	7	19	.
	Dir	1	7	48	11	11	.	47	14	9	.	42	8	20	.
	Hyp	0.5	7	61	7	2	.	49	12	9	.	42	10	18	.

Table 4: Augmenting directions found by the algorithm.

Strat	K	#Mna	#frac	#int	#orig	#fix	#upd
None	29784	55678	0.47	0.53	0.77	0.23	0.00
Alt	31361	100128	0.69	0.31	0.74	0.03	0.24
Rand	31807	120698	0.74	0.26	0.79	0.01	0.20
Dir	30805	133386	0.77	0.23	0.72	0.01	0.28
Hyp	31334	119414	0.74	0.26	0.78	0.01	0.20

better when they are used with updates because fixing variables reduces the domain of MNA and thus the space where the algorithm looks for augmentations. All in all, better directions are to be expected from a larger domain than from a restricted one.

From the aforementioned results, we conclude that the four strategies of dynamic adjustment clearly dominate the NONE strategy. The HYP update produces the best results since it lowers the proportion of unsolved problems to 14% whereas that figure is 31% for NONE. The increase of the computation time due to the update remains small compared to the static version. For large problems (ACS3–ACS5), CPLEX is unable to produce solutions as good as those that we produce within the same time limit.

6.2 Influence of the parameters

In this section, we study the influence of the parameters, i.e., of the maximum number of consecutive updates of the normalization constraint Q and the perturbation weight θ. We also give insight into the influence of the time limit. We study the influence of the parameters on the ACS3 instances, because it is sufficiently difficult and accurately portrays the overall results.

Table 5 gives results for the influence of Q. The results are shown for a random update (RAND), but we observed similar behavior for the other strategies. We chose the values of Q so that the number of MNA$^{\boldsymbol{w}}$ problems solved before variable fixing is a power of 2 ($Q = 1$, 3, 7, etc.). The time limits are almost never reached except for $Q = 31$ and $t_{\max} = 3,600$. A comparison of the last two rows of the table shows what typically happens in practice: provided the time limit is not too low, no difference is observed between the solutions. In the cases reported in Table 5, for each strategy and each level of perturbation, at most three instances reach the time limit. Furthermore, of these three cases, at most one provides a solution with a gap over 2%. Choosing $Q = 7$ is a good compromise between computational time and solution quality: for that value, only 8 instances out of 30 remain unsolved with a time limit of 1,800 s. Hence, we chose that value to test our algorithm on the benchmark.

Table 5: Influence of parameter q on performance of algorithm. The tests are performed on instance ACS3 with update strategy RAND and $\theta = 10$.

		Easy			Moderate			Hard		
Q	$t_{\max}$	$\leq 1\%$	$\leq 2\%$	$> 2\%$	$\leq 1\%$	$\leq 2\%$	$> 2\%$	$\leq 1\%$	$\leq 2\%$	$> 2\%$
1	1800	3	4	3	3	2	5	1	3	6
3	1800	4	2	4	3	4	3	2	2	6
7	1800	6	4	0	5	2	3	1	4	5
15	3600	9	1	0	4	4	2	1	2	7
31	3600	5	3	1	5	4	1	3	2	5
31	10800	5	3	1	5	4	1	3	3	4

Table 6 gives results for the influence of θ for the update strategies DIR and HYP. For these instances, the time limit was never reached. The column STRAT gives the update strategy. The results show that for DIR, $\theta = 1$ is the best value, and for HYP, $\theta = 0.5$ yields the best results. Therefore, we used these values for the global results given in Tables 3 and 4.

7 Conclusions

We have proposed a new variant of the ISUD algorithm. We modified ISUD so that the normalization of the augmentation problem is dynamically updated whenever the direction leads to a fractional direction. We have proposed four strategies to update this constraint so as to penalize fractional directions. Two of the updates (DIR and HYP) are based on theoretical observations, and the other two are based on experimental observations. We have shown that our version of the algorithm yields better results than the former version and than classical branch-and-bound techniques on a benchmark of industrial aircrew scheduling instances. The benchmark we have proposed is, to the best of our knowledge, comparable to no other from the literature. It provides large-scale instances with up to 1,700 flights and 115,000 pairings, and the instances are given in

Table 6: Influence of parameter θ on performance of algorithm. The tests are performed on instance ACS3 with $Q = 7$.

		Easy				Moderate				Hard			
STRAT	θ	$\leq 1\%$	$\leq 2\%$	$> 2\%$	t_{ISUD}	$\leq 1\%$	$\leq 2\%$	$> 2\%$	t_{ISUD}	$\leq 1\%$	$\leq 2\%$	$> 2\%$	t_{ISUD}
DIR	0.5	2	5	3	1304.74	3	5	2	1262.85	1	1	8	1627.37
	1.0	2	4	4	1374.08	3	6	1	1253.76	1	2	7	1626.40
	5.0	3	3	4	1340.40	4	4	2	1443.31	1	1	8	1673.75
	10.0	3	3	4	1346.61	4	4	2	1439.09	1	3	6	1653.87
HYP	0.5	5	4	1	1188.89	4	4	2	1200.04	1	4	5	1418.29
	1.0	4	5	1	1165.08	5	3	2	1198.11	1	4	5	1388.62
	5.0	5	2	3	1059.82	4	2	4	1156.45	1	6	3	1356.43
	10.0	4	3	3	1130.40	3	4	3	1164.54	0	4	6	1400.72

a set-partitioning form together with initial solutions that accurately mimic those of real-world applications. Our strategies allow us to find better solutions than those found by the previous version of the algorithm and to find better solutions than CPLEX finds in the same time. Our work shows the strong potential of primal algorithms for solving the crew scheduling problem, a key challenge for large airlines both financially significant and notably hard to solve.

References

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

[2] H. Bouarab, I. Elhallaoui, A. Metrane, and F. Soumis. Dynamic constraint and variable aggregation in column generation. Les Cahiers du GERAD G–2014–82, HEC Montréal, Canada, 2014.

[3] G. Desaulniers, J. Desrosiers, Y. Dumas, S. Marc, B. Rioux, M.M. Solomon, and F. Soumis. Crew pairing at Air France. European Journal of Operational Research, 97(2):245–259, 1997.

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

[5] I. Elhallaoui, A. Metrane, F. Soumis, and G. Desaulniers. Multi-phase dynamic constraint aggregation for set partitioning type problems. Mathematical Programming, 123:345–370, 2010.

[6] J.-B. Gauthier, J. Desrosiers, and M. Lübbecke. Vector space decomposition for linear programs. Les Cahiers du GERAD G–2015–26, HEC Montréal, Canada, 2015.

[7] K.L. Hoffman and M. Padberg. Solving airline crew scheduling problems by branch-and-cut. Management Science, 39(6):657–682, 1993.

[8] A. Kasirzadeh, M. Saddoune, and F. Soumis. Airline crew scheduling: Models, algorithms, and data sets. EURO Journal on Transportation and Logistics, 1–27, 2015.

[9] J. Omer, S. Rosat, V. Raymond, and F. Soumis. Improved primal simplex: A more general theoretical framework and an extended experimental analysis. INFORMS Journal on Computing, 27(4):773–787, 2015.

[10] E. Rönnberg. Contributions within two topics in integer programming: Nurse scheduling and column generation. PhD thesis, Linköping University, 2012.

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

[12] E. Rönnberg and T. Larsson. All-integer column generation for set partitioning: Basic principles and extensions. European Journal of Operational Research, 233(3):529–538, 2014.

[13] S. Rosat, I. Elhallaoui, F. Soumis, and D. Chakour. Influence of the normalization constraint on the integral simplex using decomposition. Discrete Applied Mathematics (accepted for publication), 2015.

[14] S. Rosat, I. Elhallaoui, F. Soumis, and A. Lodi. Primal cuts in the integral simplex using decomposition. Les Cahiers du GERAD G–2015–44, HEC Montréal, Canada, 2015.

[15] M. Saddoune, G. Desaulniers, and F. Soumis. Aircrew pairings with possible repetitions of the same flight number. Computers and Operations Research, 40(3):805–814, 2013.

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

[17] A. Saxena. Three articles on integral simplex method. Unpublished manuscript, OR Group, Carnegie-Mellon University, Pittsburgh, 2003.

[18] A. Schrijver. Theory of Linear and Integer Programming. John Wiley & Sons, 1998.

[19] A.S. Schulz, R. Weismantel, and G.M. Ziegler. 0/1-integer programming: Optimization and augmentation are equivalent. In P. Spirakis, editor, Algorithms – ESA '95, volume 979 of Lecture Notes in Computer Science, 473–483. Springer Berlin Heidelberg, 1995.

[20] B. Spille and R. Weismantel. Primal integer programming. In K. Aardal, G. Nemhauser, and R. Weismantel, editors, Discrete Optimization, volume 12 of Handbooks in Operations Research and Management Science, 245–276. Elsevier, 2005.

[21] M.F. Stallmann and F. Brglez. High-contrast algorithm behavior: Observation, conjecture, and experimental design. In ACM-FCRC, New York, 2007. ACM. 549075.

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

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

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