

Improving set partitioning problem solutions by zooming around an improving direction

Abdelouahab Zaghroui¹ · Issmail El Hallaoui¹ · François Soumis¹

Published online: 3 May 2018

© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract In this paper, we introduce a general framework for *vector space decompositions* that decompose the set partitioning problem into a reduced problem, defined in the vector subspace generated by the columns corresponding to nonzero variables in the current integer solution, and a complementary problem, defined in the complementary vector subspace. We show that the integral simplex using decomposition algorithm (ISUD) developed in Zaghroui et al. (Oper Res 62:435–449, 2014. <https://doi.org/10.1287/opre.2013.1247>) uses a particular decomposition, in which integrality is handled mainly in the complementary problem, to find a sequence of integer solutions with decreasing objective values leading to an optimal solution. We introduce a new algorithm using a new dynamic decomposition where integrality is handled only in the reduced problem, and the complementary problem is only used to provide descent directions, needed to update the decomposition. The new algorithm improves, at each iteration, the current integer solution by solving a reduced problem very small compared the original problem, that we define by *zooming around* the descent direction (provided by the complementary problem). This zooming algorithm is superior than ISUD on set partitioning instances from the transportation industry. It rapidly reaches optimal or near-optimal solutions for all instances including those considered very difficult for both ISUD and CPLEX.

Keywords Integral simplex · Decomposition methods · Integer programming · Set partitioning problems · Vehicle and crew scheduling

✉ Issmail El Hallaoui
issmail.elhallaoui@gerad.ca

Abdelouahab Zaghroui
abdelouahab.zaghroui@polymtl.ca

François Soumis
francois.soumis@gerad.ca

¹ Département de Mathématiques et de Génie Industriel, Polytechnique Montréal and GERAD, C.P. 6079, Succ. Centre-Ville, Montreal, QC H3C 3A7, Canada

1 Introduction

Consider the set partitioning problem (SPP) denoted by \mathbb{P} :

$$\begin{aligned} \min \quad & cx \\ (\mathbb{P}) \quad & Ax = e \\ & x_j \text{ binary, } j \in \{1, \dots, n\}, \end{aligned}$$

where A is an $m \times n$ matrix with binary entries, c is an arbitrary vector of dimension n , and $e = (1, \dots, 1)$ is a vector of dimension dictated by the context. We assume that \mathbb{P} is feasible and A is of full rank. When we relax the binary constraints (we replace them by the constraints $x_j \geq 0$), we obtain the continuous relaxation. **We denote its feasible region by LP .** It is clear that LP is bounded and consequently a polytope.

1.1 Literature review

The SPP has been widely studied in the last four decades, mainly because of its many applications in industry. A partial list of these applications includes truck deliveries (Balinski and Quandt 1964; Cullen et al. 1981), vehicle routing (Desrochers et al. 1992), bus driver scheduling (Desrochers and Soumis 1989), airline crew scheduling (Hoffman and Padberg 1993; Gamache et al. 1999; Barnhart et al. 1998; Gopalakrishnan and Johnson 2005), and simultaneous locomotive and car assignment (Cordeau et al. 2001). Several companies provide commercial optimizers to these problems using this mathematical model or one of its variants.

The literature on the SPP is abundant (see for instance Balas and Padberg 1976 (survey), Christofides and Paixao 1993; Iqbal Ali 1998). As is the case for generic integer linear programs, there are three main classes of algorithms for SPPs (Letchford and Lodi 2002): *dual fractional*, *dual integral*, and *primal methods*. Dual fractional algorithms maintain optimality and linear constraint feasibility (i.e., constraints $Ax = e$) at every iteration, and they stop when integrality is achieved. They are typically standard cutting plane procedures such as Gomory's algorithm (Gomory 1958). The classical branch-and-bound scheme is also based on a dual fractional approach, in particular for the determination of lower bounds. SPP is usually solved, especially when columns cannot be all enumerated efficiently, by branch and price (and cut) (Barnhart et al. 1998; Lubbecke and Desrosiers 2005) where each node of the branch-and-bound tree is solved by column generation. The classical approach uses the simplex algorithm, often resorting to perturbation methods to escape the degeneracy inherent to this problem, or an interior point method (such as the CPLEX barrier approach) to solve the linear relaxation of \mathbb{P} to find a lower bound. These algorithms often provide solutions that are very fractional, i.e., infeasible from the integrality point of view. This dual (or *dual fractional* as it is called in Letchford and Lodi 2002) approach strives to improve the lower bound using branching methods or cuts until a good integer solution is found. This approach is effective for small problems and remains attractive for problems of a moderate size. For large problems, a very fractional solution leads to a large branch-and-bound tree, and we must often stop the solution process without finding a good integer solution. **On** the other hand, dual integral methods, not much practical, maintain integrality and optimality, and they terminate when the primal linear constraints are satisfied. An example is the algorithm developed by Gomory (1963).

Finally, primal algorithms maintain feasibility (and integrality) throughout the process, and they stop when optimality is reached. Several authors have investigated ways to find

a nonincreasing sequence of basic integer solutions leading to an optimal solution. The existence of such a sequence was proved in Balas and Padberg (1972, 1975). The proof relies on the quasi-integrality of SPPs, i.e., every edge of the convex hull of \mathbb{P} is also an edge of LP . The proposed algorithms (Haus et al. 2001; Rönnberg and Torbjörn 2009; Thompson 2002) explore by enumeration the tree of adjacent degenerate bases associated with a given extreme point to find a basis permitting a nondegenerate pivot that improves the solution. These highly combinatorial methods are not effective for large problems, mainly because of the severe degeneracy. The number of adjacent degenerate bases associated with a given extreme point can be very huge.

Zaghroui et al. (2014) introduce an efficient algorithm, the integral simplex using decomposition (ISUD), that can find the terms of the sequence without suffering from degeneracy. ISUD is a primal approach that moves from an integer solution to an adjacent one until optimality is reached. ISUD decomposes the original problem into a reduced problem (RP) and a complementary problem (CP) that are easier to solve. We solve RP to find an improved integer solution in the vector subspace generated by columns of the current integer solution, i.e., columns corresponding to nondegenerate variables (that value 1). Note that a pivot on a negative-reduced-cost variable in this vector subspace decreases the objective value and more importantly the integrality is preserved in a straightforward manner as proved in Zaghroui et al. (2014). ISUD solves then CP to find an *integer descent direction*, i.e., leading to an improved **adjacent** integer solution, in the complementary vector subspace. Integrality is handled in both RP and CP but mainly in CP, which finds integer directions. Once the direction has been identified by CP, RP uses it to update the current integer solution and ISUD iterates until an optimal solution is reached. Such quick local improvement of integer solutions to large problems is highly desirable in practice. ISUD is more efficient than the conventional dual (fractional) approach on pure SPPs with some special structure, even though the conventional approach has been much improved since its introduction 40 years ago.

However, ISUD presents some limitations related to the facts that: (1) integrality is handled in CP where branching is not easy due to the structure of CP; (2) ISUD assumes that the polytope is quasi-integral which is restrictive (set partitioning); (3) ISUD moves to the next adjacent solution necessarily; this can increase the number of iterations. Details on the limitations of ISUD are discussed in Sect. 3.

1.2 Main contributions

To continue profiting from the advantages of such primal approach and remedy to its limitations, we introduce in Sect. 2 a new and general framework for *vector space decompositions*, simply called here RP–CP decompositions. We also provide theoretical foundations of this framework in this section. All the theoretical results discussed here are new. We particularly show that the proposed RP–CP decomposition dresses a continuum between two extremes: **On one side, the paper** (Zaghroui et al. 2014) **has** the drawback that we have to move to only adjacent solutions (the step is rather small). **On the other side, in** Balas and Padberg (1975), we cannot decompose the problem, i.e., we have to deal with the whole problem in one shot which is unpractical because of the size and severe degeneracy. The tradeoff is actually in between these two extremes. We briefly present ISUD and discuss its strengths and weaknesses in Sect. 3 and effectively show that it uses a particular RP–CP decomposition.

We introduce in Sect. 4 a new exact algorithm, called zooming algorithm, using a dynamic RP–CP decomposition where, in contrast to ISUD, integrality is handled only in RP, and CP is only used to provide descent directions to update the decomposition. When CP finds a

fractional descent direction, i.e., leading to a fractional solution, instead of using cuts (Rosat et al. 2014b, 2017) or branching (Zaghrouti et al. 2014) to find an integer direction, we use this descent direction to indicate an area of potential improvement that is likely to contain an *improved integer solution*. The new algorithm then constructs a very small RP as a MIP by zooming around the descent direction. The RPs solved are ten times smaller than the original problem and very easy to solve. The zooming algorithm is globally primal (it moves through a decreasing sequence of integer solutions to the optimal solution of \mathbb{P}) but locally dual fractional, i.e., when necessary it solves a small (local) MIP using the dual fractional approach. This allows to solve large industrial SPPs within an exact primal paradigm. We also provide some new theoretical insights to explain why the algorithm is efficient.

Numerical results in Sect. 5 show that this new algorithm reaches all the time optimal or near optimal solutions on some very hard (to both ISUD and CPLEX) vehicle and crew scheduling problems without increasing the solution time. In Sect. 6, we discuss possible extensions of the approach.

2 RP–CP decomposition

We present in this section the RP–CP decomposition as a general framework. We discuss its basics and the descent directions that shape it. We end up this section with a characterization of an optimal decomposition leading to an optimal solution.

2.1 Decomposition basics

Let \mathcal{Q} be an index subset of linearly independent columns of A , containing at least indices of columns corresponding to positive variables in a given solution to \mathbb{P} . We partition the columns of A into two groups, those that are compatible with \mathcal{Q} and those that are not, using the following definition:

Definition 2.1 A column (and its corresponding variable) or a convex combination of columns is said to be compatible with \mathcal{Q} if it can be written as a linear combination of columns indexed in \mathcal{Q} . Otherwise, it is said incompatible.

The index set of compatible columns is denoted $\mathcal{C}_{\mathcal{Q}}$ and the index set of incompatible columns is denoted $\mathcal{I}_{\mathcal{Q}}$. Clearly $\mathcal{Q} \subseteq \mathcal{C}_{\mathcal{Q}}$ and $|\mathcal{Q}| \leq m$. \mathcal{Q} is said to be **nondegenerate** if it is restricted to columns corresponding to positive variables (in a given solution); otherwise it is said **degenerate**.

Let $A_{\mathcal{Q}} = \begin{pmatrix} A_{\mathcal{Q}}^1 \\ A_{\mathcal{Q}}^2 \end{pmatrix}$ be a submatrix of A composed of columns indexed in \mathcal{Q} where $A_{\mathcal{Q}}^1$ is, without loss of generality, composed of the first $|\mathcal{Q}|$ linearly independent rows. $A_{\mathcal{Q}}^2$ is of course composed of dependent rows. Similarly, $A_{\mathcal{C}_{\mathcal{Q}}} = \begin{pmatrix} A_{\mathcal{C}_{\mathcal{Q}}}^1 \\ A_{\mathcal{C}_{\mathcal{Q}}}^2 \end{pmatrix}$ (resp. $A_{\mathcal{I}_{\mathcal{Q}}} = \begin{pmatrix} A_{\mathcal{I}_{\mathcal{Q}}}^1 \\ A_{\mathcal{I}_{\mathcal{Q}}}^2 \end{pmatrix}$) is a submatrix of A composed of columns indexed in $\mathcal{C}_{\mathcal{Q}}$ (resp. $\mathcal{I}_{\mathcal{Q}}$) where $A_{\mathcal{C}_{\mathcal{Q}}}^1$ and $A_{\mathcal{I}_{\mathcal{Q}}}^1$ are also composed of the same first $|\mathcal{Q}|$ rows as in $A_{\mathcal{Q}}^1$ which means that the matrix A can be decomposed as follows: $A = \begin{pmatrix} A_{\mathcal{C}_{\mathcal{Q}}}^1 & A_{\mathcal{I}_{\mathcal{Q}}}^1 \\ A_{\mathcal{C}_{\mathcal{Q}}}^2 & A_{\mathcal{I}_{\mathcal{Q}}}^2 \end{pmatrix}$.

When we consider only compatible columns c_Q and therefore the first $|Q|$ rows, we obtain the RP_Q , which can be formulated as

$$(RP_Q) \min_{x_{c_Q}} c_{c_Q} \cdot x_{c_Q} \tag{2.1}$$

$$\text{s.t. } A_{c_Q}^1 x_{c_Q} = e \tag{2.2}$$

$$x_j \text{ binary, } j \in c_Q \tag{2.3}$$

where c_{c_Q} is the subvector of the compatible variables costs and x_{c_Q} is the subvector of compatible variables. RP_Q is therefore an SPP restricted to the compatible variables and the first $|Q|$ rows. RP_Q is equivalent to RP because the other constraints are dependent. Observe that its columns are in $\mathbb{R}^{|Q|}$ (a reduced dimension). It is **important** to stress the fact that each solution to RP_Q can be completed by zeros to form a solution to \mathbb{P} .

The CP (containing the incompatible variables) is formulated as follows:

$$(CP_Q) \quad z_Q^{CP} = \min_{v, \lambda} \sum_{j \in I_Q} c_j v_j - \sum_{l \in Q} c_l \lambda_l \tag{2.4}$$

$$\text{s.t. } \sum_{j \in I_Q} v_j A_j - \sum_{l \in Q} \lambda_l A_l = 0 \tag{2.5}$$

$$e \cdot v + u = 1 \tag{2.6}$$

$$v \geq 0, u \geq 0, \lambda \in \mathbb{R}^{|Q|}. \tag{2.7}$$

The positive v_j variables indicate entering variables and the positive λ_l variables indicate exiting variables. We look for a group of entering variables that will replace the exiting variables. Of course, the cost difference (2.4) between the entering and exiting variables (the reduced cost) must be negative for a minimization problem in order to improve the objective value. In other words, we look, by imposing constraints (2.5), for a convex combination, with a negative reduced cost, of incompatible columns A_j (of the constraint matrix A) that is compatible with Q according to Definition 2.1, i.e., that is a linear combination of columns A_l indexed in Q . Without constraint (2.6), the feasible domain of CP_Q is an unbounded cone and the solution is unbounded. With this constraint, the problem is bounded and provides a normalized improving direction. It is therefore called a *normalization constraint*. To guarantee feasibility, we add an artificial variable u that costs 0 and only contributes to the normalization constraint (by 1); this way, CP_Q is feasible and $z_Q^{CP} \leq 0$. The constraints (2.7) define the feasible domain for the variables. *Observe that the integrality constraints are kept in RP only.*

Since the variables λ_l , linked to variables v_j via constraints 2.5, are associated with linearly independent columns $A_l, l \in Q$, they can be substituted by the variables $v_j, j \in I_Q$. Actually, we have $\lambda = (A_Q^1)^{-1} A_{I_Q}^1 v$. After the substitution, the CP can be rewritten in the following equivalent form:

$$(CP_Q) \quad z_Q^{CP} = \min_v \tilde{c} \cdot v \tag{2.8}$$

$$\text{s.t. } MA_{I_Q} v = 0 \tag{2.9}$$

$$\text{Constraints (2.6)–(2.7)} \tag{2.10}$$

where $v \in \mathbb{R}^{|I_Q|}, \tilde{c} = (c_{I_Q}^T - c_Q^T (A_Q^1)^{-1} A_{I_Q}^1)$ is the vector of reduced costs with respect to the constraints of the RP and $M = (A_Q^2 (A_Q^1)^{-1}, -I_{m-|Q|})$ is a projection matrix, called the *compatibility matrix*, on the complementary vector subspace. c_{I_Q} is the subvector of incompatible variables costs and $I_{m-|Q|}$ is the identity matrix of dimension $m - |Q|$. With

the normalization constraint, the dimension of the columns of the CP is $m - |\mathcal{Q}| + 1$, i.e., the dimension of the complementary vector subspace + 1.

We can classify incompatible columns according to their *incompatibility degree*. The incompatibility degree can be mathematically defined as “the distance” from column A_j to the vector subspace generated by the columns indexed by \mathcal{Q} . Of course, if A_j is in this vector subspace, i.e., it is a linear combination of columns indexed by \mathcal{Q} ; this distance will be 0, indicating that the column is compatible. If the distance (incompatibility degree) is positive, the column is incompatible. The incompatibility degree plays a crucial role in the partial pricing strategy of Sect. 4.2. In this paper, we compute the incompatibility degree of a column A_j as the number of nonzeros of MA_j . A column A_j and its associated variable x_j are said to be *k-incompatible* with respect to \mathcal{Q} if its degree of incompatibility is equal to k . Compatible columns and variables are called *0-incompatible*.

If we relax the binary constraints (2.3), we obtain the improved primal simplex decomposition introduced in El Hallaoui et al. (2011). In that paper, the authors proved that if \mathcal{Q} is nondegenerate; a pivot on any negative-reduced-cost compatible variable or a sequence of pivots on the set of the entering variables (i.e., $v_j > 0$) improves (strictly) the current solution of LP iff $z_{\mathcal{Q}}^{CP} < 0$.

2.2 Improving directions

Let x be the current integer solution to \mathbb{P} and $s_{CP_{\mathcal{Q}}} = (v, \lambda, u)$ an optimal solution to $CP_{\mathcal{Q}}$. We have shown in Rosat et al. (2017) that $d_{CP_{\mathcal{Q}}} = (v, -\lambda, 0)$, defined from $s_{CP_{\mathcal{Q}}}$ and completed by $|\mathcal{C}_{\mathcal{Q}} \setminus \mathcal{Q}|$ zeros (corresponding to compatible variables that are absent from $CP_{\mathcal{Q}}$) to have the same dimension as x , is a descent direction if $u = 0$ of course, meaning that $z_{\mathcal{Q}}^{CP} < 0$. We have shown also in Rosat et al. (2017) that there exists $\rho > 0$ such that $d = \rho d_{CP_{\mathcal{Q}}}$ is an extremal descent direction, in a sense that $x' = x + d$ is an extreme point of LP. In other words, we can always build extremal directions from the optimal solutions to CP.

Definition 2.2 $s_{CP_{\mathcal{Q}}}$ and its associated descent direction d are said to be *disjoint* if the columns $\{A_j | v_j > 0, j \in I_{\mathcal{Q}}\}$ are pairwise row-disjoint.

Definition 2.3 $s_{CP_{\mathcal{Q}}}$ and its associated descent direction d are said to be *integer* if $x + d$ is an improved integer solution and fractional otherwise.

The following proposition provides a general characterization of the integrality of descent directions. Let $\mathcal{Q}_{int} = \{j : x_j = 1\} \subset \mathcal{Q}$ be the set of indices of columns of the current integer solution.

Proposition 2.4 $d_{CP_{\mathcal{Q}}}$ is integer \iff

1. $\{A_j | v_j > 0, j \in I_{\mathcal{Q}}\} \cup \{A_\ell | \lambda_\ell < 0, \ell \in \mathcal{Q}\}$ is a set of pairwise row-disjoint columns.
2. $\{l | \lambda_\ell > 0, \ell \in \mathcal{Q}\} \subset \mathcal{Q}_{int}$.
3. $\{l | \lambda_\ell < 0, \ell \in \mathcal{Q}\} \subset \mathcal{Q} \setminus \mathcal{Q}_{int}$.

Proof First (\implies), if the direction found by $CP_{\mathcal{Q}}$ is integer, this means that there exists a $\rho > 0$ such that $x' = x + d$ is an integer solution, with $d = \rho \delta$ and $\delta = d_{CP_{\mathcal{Q}}}$. For each j , we have $\delta_j = \frac{x'_j - x_j}{\rho}$. As x and x' are 0–1 vectors, we have:

- $x'_j = x_j \iff \delta_j = 0$
- $x'_j = 1, x_j = 0 \iff \delta_j = \frac{1}{\rho}$

- $x'_j = 0, x_j = 1 \iff \delta_j = \frac{-1}{\rho}$

We have $\sum_{j \in I_Q} v_j A_j - \sum_{l \in Q} \lambda_l A_l = 0$. So, we have $\delta_j = v_j$ for $j \in I_Q$ and $\delta_l = -\lambda_l$ for $l \in Q$. We are interested in nonzero components (support) of δ . Remark that $\lambda_l > 0$ ($\delta_j = -\lambda_l = \frac{-1}{\rho}$) is equivalent to $x_j = 1$ and $x'_j = 0$ which means that $\{l | \lambda_l > 0, l \in Q\} \subset Q_{int}$. Observe also that $\lambda_l < 0$ ($\delta_j = -\lambda_l = \frac{1}{\rho}$) is equivalent to $x_j = 0$ and $x'_j = 1$. That means that $\{l | \lambda_l < 0, l \in Q\} \subset Q \setminus Q_{int}$. We can rewrite (2.5) as

$$\sum_{j: j \in I_Q, v_j > 0} A_j + \sum_{l: \lambda_l < 0} A_l = \sum_{l: \lambda_l > 0} A_l$$

From above, we can conclude that $\{A_j | v_j > 0, j \in I_Q\} \cup \{A_l | \lambda_l < 0, l \in Q\}$ is a set of pairwise row-disjoint columns.

Second, to prove the left implication (\Leftarrow), we know that:

$$\sum_{j | j \in I_Q, v_j > 0} v_j A_j - \sum_{l | \lambda_l < 0} \lambda_l A_l = \sum_{l | \lambda_l > 0} \lambda_l A_l$$

Observe that both the columns of the right and left sides are pairwise row-disjoint and consequently they are linearly independent. We can show that there exists a certain positive ρ such that $v_j = \frac{1}{\rho}, \lambda_l = \frac{-1}{\rho}$ when $\lambda_l < 0$ and $\lambda_l = \frac{1}{\rho}$ when $\lambda_l > 0$ is a unique solution (involving columns A_j corresponding nonzero variables v_j and λ_l) to the linear system just above. So, the solution x' obtained by replacing in x the columns of the right side by the columns of the left side is an improved integer solution because the columns are disjoint and the cost difference between the two solutions is negative. \square

The proposition 2.4 is a new theoretical result that generalizes: (1) the result of Zaghroui et al. (2014) showing that if the solution to the CP, when Q is nondegenerate (i.e., $Q = Q_{int}$), is disjoint then we obtain an integer descent direction, and (2) the result of Balas and Padberg (1975) working with sets Q containing all degenerate basic variables (containing indices of all basic variables including degenerate ones). This proposition is interesting because it shows that the proposed RP–CP decomposition dresses a continuum between two extremes: Zaghroui et al. (2014) on one side, with the drawback that we have to move to only adjacent solutions (the step is rather small) and Balas and Padberg (1975) in the other side, where we cannot decompose the problem, i.e., we have to deal with the whole problem in one shot which is unpractical because of the size and severe degeneracy. The tradeoff is actually in between these two extremes. This proposition means that unlike ISUD, a disjoint d_{CP_Q} does not guarantee that x' is integer in the general case where Q is degenerate. This proposition is also interesting because it characterizes integral directions and proposes a possible way (not implemented yet) of strengthening the CP formulation as noticed just below:

Remark 2.5 Observe also that:

- The formulation (2.8)–(2.10) of CP_Q can be strengthened by imposing that $\lambda_l \geq 0, l \in Q_{int}$ and $\lambda_l \leq 0, l \in Q \setminus Q_{int}$ where $\lambda = (A^1_Q)^{-1} A^1_{I_Q} v$.
- The strengthened formulation is still bounded and the improved solution x' is not necessarily adjacent to x .

2.3 Optimality characterization

The following proposition proves the existence of an optimal RP–CP decomposition, in the sense that instead of solving the original problem, we may solve smaller problems (the RP

and CP) and obtain an optimal solution to the original problem. This theoretically motivates further our research and provides in some sense a mathematical foundation for the proposed approach.

Proposition 2.6 *There exists an optimal decomposition RP–CP such that an optimal solution to $RP_{\mathcal{Q}}$ is also optimal to \mathbb{P} and $z_{\mathcal{Q}}^{CP}$, the objective value of $CP_{\mathcal{Q}}$, is nonnegative.*

Proof Consider in \mathcal{Q} all columns of the optimal solution and in RP all columns that are involved in fractional descent directions. It is sufficient (at least theoretically) to add a finite number of facets to the RP to cut the fractional solutions. Obviously, the CP cannot find, after adding these cuts, a descent direction, so $z_{\mathcal{Q}}^{CP} \geq 0$. We would like to outline that adding the cuts in the same way as in Rosat et al. (2014b) (see specifically Propositions 6–8) does not change anything to CP. \square

This decomposition allows handling the integrality constraints in the RP instead of handling them, as ISUD does, in the CP. In the worst case, the RP may coincide with the original problem. In practice, large SPPs are generally highly degenerate, and the RP is significantly smaller than the original problem as a result of this inherent degeneracy.

Below, we present two propositions that help proving the exactness of the proposed approach and assessing the solution quality. To prove these propositions, we need the following lemma. We suppose that $RP_{\mathcal{Q}}$ and $CP_{\mathcal{Q}}$ are solved to optimality. Let π be the vector of dual values associated with constraints 2.5 and y the dual value associated with constraint 2.6.

Lemma 2.7 *Let $\bar{c}_j = c_j - \pi^T A_j$ be the reduced cost of variable x_j . We have $\bar{c}_j \geq z_{\mathcal{Q}}^{CP}$, $j \in \{1, \dots, n\}$ and $\bar{c}_j = z_{\mathcal{Q}}^{CP}$ for variables such that $v_j > 0$.*

Proof The dual to $CP_{\mathcal{Q}}$, denoted $DCP_{\mathcal{Q}}$, is

$$(DCP_{\mathcal{Q}}) \quad z_{\mathcal{Q}}^{DCP} = \max_{\pi, y} y \tag{2.11}$$

$$\text{s.t.} \quad c_l - \pi^T A_l = 0 \quad l \in \mathcal{Q} \tag{2.12}$$

$$c_j - \pi^T A_j \geq y \quad j \in I_{\mathcal{Q}} \tag{2.13}$$

$$y \in \mathbb{R}, \pi \in \mathbb{R}^m. \tag{2.14}$$

So, $z_{\mathcal{Q}}^{CP} = z_{\mathcal{Q}}^{DCP}$ and represents the maximum value of y (the minimum reduced cost). \square

The next proposition is a new result that generalizes the special cases proved in Zaghrouti et al. (2014) where $\mathcal{Q} = \mathcal{Q}_{int}$ and in El Hallaoui et al. (2011) where integrality constraints (2.3) are not considered.

Proposition 2.8 *If the optimal solution to $RP_{\mathcal{Q}}$ is not optimal to \mathbb{P} , then $z_{\mathcal{Q}}^{CP} < 0$.*

Proof The idea of the proof is simple. Suppose that $z_{\mathcal{Q}}^{CP} \geq 0$ and $RP_{\mathcal{Q}}$ is solved to optimality by adding facets like in the proof of Proposition 2.6. By Lemma 2.7, all variables will have nonnegative reduced costs. That means that the actual solution is optimal to \mathbb{P} , which is a contradiction. \square

The following proposition provides a lower bound on the optimal objective value of \mathbb{P} using the RP–CP decomposition. Such a lower bound can be found by solving the linear relaxation of the problem, but this can be computationally more expensive. We instead find a good lower bound using the information provided by the RP and CP.

Proposition 2.9 *Let \bar{z} be the current optimal objective value of $RP_{\mathcal{Q}}$ and z^* the value of an optimal solution to \mathbb{P} . We have $\bar{z} + m \cdot z_{\mathcal{Q}}^{CP} \leq z^* \leq \bar{z}$.*

Proof By Lemma 2.7, the reduced costs are lower bounded by $z_{\mathcal{Q}}^{CP}$. The best improvement from entering a variable into the basis with a full step (not larger than 1) is $z_{\mathcal{Q}}^{CP}$, and the maximum number of nonzero variables in an optimal basis of \mathbb{P} is m . Thus, the objective value cannot be improved by more than $m \cdot z_{\mathcal{Q}}^{CP}$. \square

We can easily build an example where this bound is reached. Consider m columns having their reduced cost equal to $z_{\mathcal{Q}}^{CP}$ and forming an identity matrix. These columns can simultaneously enter the basis, and the objective value will be $\bar{z} + m \cdot z_{\mathcal{Q}}^{CP}$. We notice that in practice we generally know a priori the maximum number of columns of the solution (drivers, pilots) that is 10–20 times less than the number of constraints m . This tightens the bound. This bound can be used as an indicator of the quality of the integer solution of cost \bar{z} . This is particularly useful for stopping the solving process when the current solution quality becomes acceptable. A similar bound has been shown good in Bouarab et al. (2017) for a different context (context of column generation for solving the LP). Observe that if $z_{\mathcal{Q}}^{CP} = 0$, then $\bar{z} = z^*$ because $\bar{z} \leq z^* \leq \bar{z}$. The lower bound varies with \mathcal{Q} . The idea is to find a tradeoff for \mathcal{Q} between two extremes \mathcal{Q}_{int} and $\{1, \dots, m\}$ such that the lower bound provided by Proposition 2.9 is good enough to use as a criterion to stop the solution process and $RP_{\mathcal{Q}}$ is easy to solve.

3 Integral simplex using decomposition (ISUD)

Suppose that we have an integer basic solution to \mathbb{P} with p positive variables. ISUD uses a particular RP–CP decomposition where (1) \mathcal{Q} is nondegenerate, i.e., $\mathcal{Q} = \mathcal{Q}_{int}$, and (2) we add integrality constraints to the CP: its solution must be integer according to Definition 2.3. We thus obtain what we refer to as $CP_{\mathcal{Q}}^I$, a CP with disjoint column requirement; $CP_{\mathcal{Q}}$ is its relaxation.

ISUD starts by solving $RP_{\mathcal{Q}}$. We can solve $RP_{\mathcal{Q}}$ with any commercial MIP solver, but a simpler approach is as follows. Observe that pivoting on any negative-reduced-cost compatible variable improves the objective value of \mathbb{P} and thus yields an integer descent direction because $RP_{\mathcal{Q}}$ is nondegenerate. If we cannot improve the solution of $RP_{\mathcal{Q}}$ with compatible variables, ISUD solves $CP_{\mathcal{Q}}^I$ to get a group of (more than one) entering variables yielding an integer descent direction. The ISUD algorithm can be stated as follows:

Step 1: Find a good initial heuristic solution x_0 and set $x = x_0$.

Step 2: Get an integer descent direction d either by pivoting on a negative-reduced-cost compatible variable of $RP_{\mathcal{Q}}$ or by solving $CP_{\mathcal{Q}}^I$.

Step 3: If no descent direction is found then stop: the current solution is optimal. Otherwise, set $x = x + d$, update \mathcal{Q} , and go to Step 2.

The key ISUD findings are the following:

- From an integer solution, a pivot on a negative-reduced-cost compatible variable of $RP_{\mathcal{Q}}$ produces an improved integer solution (in Step 2).
- The sequence of pivots of $RP_{\mathcal{Q}}$ on entering variables identified by $CP_{\mathcal{Q}}^I$ provides an improved integer solution. This result is a corollary of Proposition 2.4.
- The $RP_{\mathcal{Q}}$ and $CP_{\mathcal{Q}}^I$ improvements are sufficient to achieve optimality in Step 3.
- $CP_{\mathcal{Q}}$ produces disjoint solutions 50–80% of the time, without any branching. The normalization constraint (2.6) plays an important role in the efficiency of ISUD: it favors

integrality and helps $CP_{\mathcal{Q}}^I$ to find disjoint solutions with relatively few columns. In fact, ISUD focuses on making a series of easy improvements without intensive combinatorial searching, which is generally caused by a difficult branch and bound.

- Optimal solutions have been obtained for crew assignment problems with up to 570,000 variables, usually in minutes.

ISUD is a promising approach but it has some limitations. First, if $CP_{\mathcal{Q}}$ fails to find an integer direction, its structure cannot easily be exploited (by commercial solvers) for branching or cutting purposes, because of the normalization constraint. $CP_{\mathcal{Q}}^I$ needs a sophisticated specialized branching scheme (see Zaghroui et al. 2014, Subsection 3.3 for more details). The tests in Zaghroui et al. (2014) use a heuristic implementation of the branching scheme: a deep branching only where the branch called “0-branch” sets all variables with $v_j > 0$ in the fractional descent direction to 0 and tries to find a completely different descent direction that is integer. There is no guarantee that this implementation will find an optimal integer solution. In fact, ISUD finishes far from the optimal solution up to 40% of the time on difficult instances. Second, the improvement per iteration in the objective value of the RP is rather small because we move from one extreme point to an adjacent one. Third, ISUD cannot directly handle additional linear (non-SPP) constraints because the quasi-integrality property may be lost in the presence of such constraints. There is clearly room for improvement to ISUD. We present in the following section a new algorithm better than ISUD.

4 Zooming around an improving direction

In this section, we present an exact zooming approach using a more flexible RP–CP decomposition to resolve the issues raised above about ISUD. Since the optimal RP–CP decomposition is not known a priori, we propose an iterative algorithm to find it in a primal paradigm, i.e., while finding a nonincreasing sequence of integer solutions. We start with an initial decomposition that we iteratively update to better approximate an optimal decomposition. The mechanics of the zooming algorithm are discussed in detail in the next two subsections.

4.1 Zooming algorithm

The zooming pseudocode is provided below. We follow by presenting its main steps and giving some insights that explain why this algorithm is efficient.

Step 1: Find a good heuristic initial solution x_0 and set $x = x_0$, $\mathcal{Q} = \mathcal{Q}_{int}$, $d = 0$.

Step 2: Find a better integer solution around d :

- Increase \mathcal{Q} : set $\mathcal{Q} = \mathcal{Q} \cup \{j : d_j > 0\}$.
- Construct and solve $RP_{\mathcal{Q}}$.
- Update x and \mathcal{Q} : if x is improved, set $\mathcal{Q} = \mathcal{Q}_{int}$.

Step 3: Get a descent direction d :

- Solve $CP_{\mathcal{Q}}$ to get a descent direction d .
- If no descent direction can be found or $|z_{\mathcal{Q}}^{CP}|$ is small enough then stop: the current solution is optimal or near optimal.
- Otherwise, go to Step 2.

In Step 1, we need an initial integer solution x_0 to start up the algorithm; it can be found by a heuristic technique. The initial solution may contain artificial variables to be feasible. In

some cases, for example when reoptimizing after a perturbation (such as some flight cancellation), the initial solution could simply be the currently implemented solution. Generally, the reoptimized solution deviates slightly from the initial one, which is interestingly preferred in practice. This avoids errors and additional managerial costs. This initial solution is used to define the initial RP–CP decomposition.

In Step 2, we increase \mathcal{Q} by the incompatible variables having positive d_j value. We populate $RP_{\mathcal{Q}}$ with variables (columns) that are compatible with the new \mathcal{Q} and we solve it using the partial pricing strategy described in Sect. 4.2. $RP_{\mathcal{Q}}$ is of small size and good properties as discussed below and can be solved efficiently by any MIP solver. We can easily show the next proposition:

Proposition 4.1 *By increasing \mathcal{Q} in Step 2, the current solution x remains an integer solution to $RP_{\mathcal{Q}}$. Furthermore, $x+d$ is a basic feasible solution (extreme point) to the linear relaxation of $RP_{\mathcal{Q}}$.*

Proof The proof is quite simple. By increasing \mathcal{Q} , we ensure that variables with positive values x_j and d_j are still compatible with the new \mathcal{Q} . So, they will be present in $RP_{\mathcal{Q}}$. \square

We can use $x+d$ to provide the MIP solver of $RP_{\mathcal{Q}}$ with an initial feasible basis, thus avoiding a phase I in the simplex algorithm. So, the solution of $CP_{\mathcal{Q}}$ is useful even if d is fractional. Also, x can serve as an upper bound. The MIP solver may use x to eliminate some useless branches ($RP_{\mathcal{Q}}$ has smaller gap as discussed in Sect. 4.2) or to find an improved integer solution using heuristics.

Because x and $x+d$ are solutions to the linear relaxation of $RP_{\mathcal{Q}}$, we can say that $RP_{\mathcal{Q}}$ is actually a neighborhood around the improving direction d heading from x to $x+d$. Observe that at least an improved integer solution is adjacent to x since SPP is quasi-integral, if x is of course not optimal. Hence, such a neighborhood is likely to encompass an improved integer solution. We emphasize that $x+d$ is adjacent to x since d is minimal (See El Hallaoui et al. 2011). Consequently, the variables of the $CP_{\mathcal{Q}}$ solution cover a small portion of the constraints. We therefore do not increase \mathcal{Q} too much and $RP_{\mathcal{Q}}$ remains tractable. We thus zoom around an off-center neighborhood in the direction of lower-cost solutions as illustrated in Fig. 1.

If d is integer, we do not need a MIP solver. We can easily update the current integer solution x by replacing it by $x+d$. In any case, whenever we obtain a better integer solution we reduce $RP_{\mathcal{Q}}$ by decreasing \mathcal{Q} : we redefine it according to this better integer solution implying fewer disjoint columns. We thus avoid increasing indefinitely the size of the RP.

In Step 3, we solve $CP_{\mathcal{Q}}$ to get a descent direction d . The direction d is used to guide the search in Step 2 to an area of potential improvement. The direction d is obtained as explained in Sect. 2: $d_j = \rho v_j$ for $j \in I_{\mathcal{Q}}$, $d_j = -\rho \lambda_j$ for $j \in \mathcal{Q}$, and $d_j = 0$ otherwise. The constant ρ is computed such that $x+d$ is an extreme point of LP. If $z_{\mathcal{Q}}^{CP} \geq 0$ the current solution is optimal and obviously an optimal decomposition is reached. If $|z_{\mathcal{Q}}^{CP}| \leq \frac{\epsilon}{m}$, the current solution is close to optimality according to Proposition 2.9. ϵ is a predetermined threshold. The CP is easy to solve using the partial pricing strategy discussed in the next subsection. The next proposition discusses the convergence of the algorithm.

Proposition 4.2 *The zooming algorithm provides an optimal solution to \mathbb{P} in a finite number of iterations.*

Proof The objective value decreases, which means that the solution changes, if $RP_{\mathcal{Q}}$ improves the current integer solution in Step 2 either by updating it using an integer direction

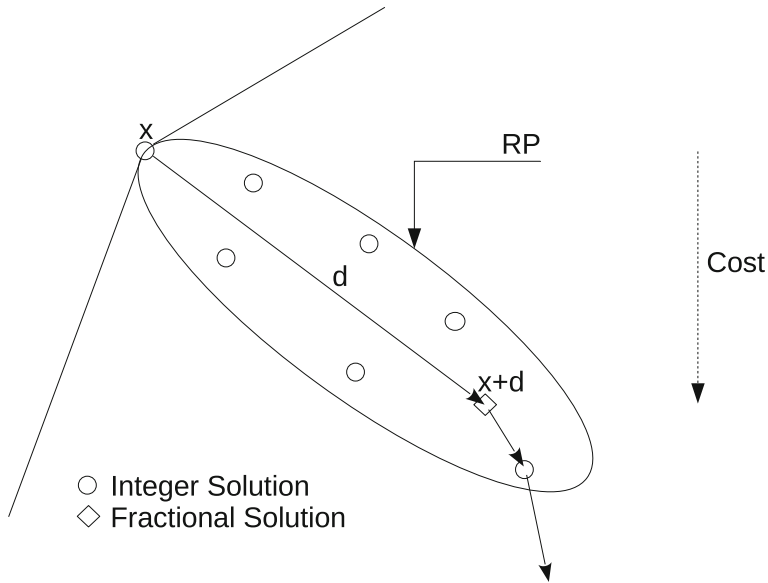


Fig. 1 Zooming around an improving direction

d or by solving the RP_Q using a MIP solver. If it is fractional, we increase Q . In this case, the number of rows of RP_Q is increased by at least one. As long as the current integer solution is not optimal to \mathbb{P} , CP_Q will find a descent direction as claimed by Proposition 2.8. Note that while we do not improve the current integer solution and CP_Q succeeds in generating descent directions, we increase Q . In the worst case, the RP_Q will coincide with the original problem after at most m iterations, where m is the number of constraints. The number of iterations can be upper bounded by m times the number of different solutions of \mathbb{P} which is itself bounded by 2^n . □

4.2 Partial pricing strategy

We propose solving RP_Q and CP_Q through a sequence of phases by performing a partial pricing based on the incompatibility degree. This latter is used to define the phase number as follows:

Definition 4.3 RP_Q and CP_Q are said to be in phase k when only the variables that are q -incompatible with $q \leq k$ are priced out; k is called the phase number.

We use a predetermined sequence of phases with increasing numbers to accelerate the solution of RP_Q in Step 2 and CP_Q in Step 3. This generalizes the multiphase strategy introduced in El Hallaoui et al. (2010). Each phase corresponds to a different level of partial pricing. If RP_Q or CP_Q fails in phase k , we go to the subsequent phase. To ensure the exactness of the algorithm, we end with a phase where all the variables (resp. compatible variables) are priced out for CP_Q (resp. RP_Q). The advantages of the multiphase strategy are discussed below and confirmed by the experiments reported in Sect. 5. In the remainder of this paper, we use the following definition of the gap:

Definition 4.4 For a given problem, the **current (integrality) gap** refers to the gap between the current integer solution and its LP solution.

Proposition 4.5 *We have:*

- a. *The current gap of $RP_{\mathcal{Q}}$ in phase k is smaller than the current gap of the original problem.*
- b. *The current gap of $RP_{\mathcal{Q}}$ increases with the phase number.*

Proof a. It is easy to see that $RP_{\mathcal{Q}}$ is a restriction of \mathbb{P} . Therefore, the current gap of $RP_{\mathcal{Q}}$ is smaller than the current gap of \mathbb{P} .

- b. By Definition 4.3, all the k -incompatible columns are present in $RP_{\mathcal{Q}}$ in phase $k + 1$. Therefore, $RP_{\mathcal{Q}}$ in phase k is a restriction of $RP_{\mathcal{Q}}$ in phase $k + 1$, and it follows that the current gap of $RP_{\mathcal{Q}}$ increases with the phase number. □

This proposition suggests starting with lower phases and increasing the phase number as necessary, because it is much easier to solve the $RP_{\mathcal{Q}}$ using an MIP solver when the gap is small. This is consistent with this primal approach that locally improves the current integer solution.

We have shown in Bouarab et al. (2017) (See Proposition 4.8) that the number of nonzeros of the $CP_{\mathcal{Q}}$ constraint matrix in phase k is less than $(k + 1) \times q_k$, where q_k is the number of columns present in $CP_{\mathcal{Q}}$ in this phase. The number of nonzeros of the $RP_{\mathcal{Q}}$ constraint matrix in phase k is also small. The density of $CP_{\mathcal{Q}}$ and $RP_{\mathcal{Q}}$ depends much more on the phase number and on how we increase \mathcal{Q} than on the density of the original problem. Recall that the number of rows in $RP_{\mathcal{Q}}$ is $|\mathcal{Q}|$ because columns indexed by \mathcal{Q} are linearly independent and we remove dependent rows from $RP_{\mathcal{Q}}$, thus reducing its dimension.

This shows that $RP_{\mathcal{Q}}$ and $CP_{\mathcal{Q}}$ are easy to solve, especially in the lower phases. In practice we reach optimality after four or five phases (on average) for many vehicle and crew scheduling problems. The number of nonzeros per column would generally not exceed 4-5 in practice (instead of 40 for some instances of the original problem). For instance, an $RP_{\mathcal{Q}}$ in a lower phase with a low density is relatively easy to solve by commercial solvers such as CPLEX. We can use the branching and cutting of such commercial solvers locally and effectively on SPPs that are thirty times or more smaller than the original problem. Commercial solvers are known to be efficient on small to moderate SPPs; they can quickly improve the objective value, if not optimal yet.

5 Experimentation

The goal of our tests is to compare the performance of the zooming algorithm (ZOOM) with that of ISUD. We did our tests on a 2.7 Ghz i7-2620M Linux machine. The solver is CPLEX 12.4; it is used to solve the RP as an MIP and the CP as an LP.

5.1 Instances

We use the same instances that were used to test ISUD (Zaghrouti et al. 2014) against CPLEX. The instances are in three categories: small (800rows \times 8900 columns), medium (1200 \times 139,000), or large (1600 \times 570,000). Small (aircrew scheduling) instances are the largest, in terms of the number of constraints, and the most “difficult” in the OR Library test bed. Medium and large (driver and bus scheduling) instances are difficult. The related problem and parameters are exactly the same described in Haase et al. Haase et al. (2003). In total, we have 90 instances, 30 instances in each category.

Their accompanying initial solutions were produced using a perturbation process as explained in Zaghrouti et al. (2014). The instance difficulty increases as the so-called

Table 1 CPLEX: average results

	Time	Init.	Best int.	LP value	LP time	Int. opt
Small	7	316,528.0	56,137	55,535.4	0.4	56,137
Medium	600	4,066,595.2	3,648,462	2,450,953.7	6.5	2,451,145
Large	3600	5,475,541.0	5,287,448	3,301,655.5	46	3,302,462

unperturbed ratio decreases. The authors developed in Zaghrouti et al. (2014) a simulation technique to obtain a feasible initial solution with a certain level of good primal information similar to the initial solutions generally available in practice for crew scheduling problems. We measure the primal information by the percentage of consecutive pairs of tasks in the initial solution that remain grouped in the optimal solution. In vehicle and crew scheduling problems, many tasks that are consecutive in the vehicle routes are also consecutive in the crew schedules. We observe the same property when we reoptimize a planned solution after perturbation, for example, after some flight cancellation.

Given an SPP instance with a known optimal integer solution, the perturbation process produces a new instance with the same optimal integer solution and an initial integer solution that is a certain distance from the optimal one. It does this by randomly cutting and recombining the schedules (columns) of the known solution.

The perturbation process particularly simulates the perturbed planned solutions (flight cancellation for instance) and consists of randomly selecting two of the columns in the (planned) solution and replacing them with two different columns covering the same tasks. It chooses two tasks belonging to two different columns in the solution and connects them to create a new column. This process is repeated until the number of unchanged columns is below a certain percentage of the total number of columns in the solution. This parameter is set to 50, 35, and 20% for *low*, *moderate*, and *severe* perturbation. The newly generated columns are added to the problem and given a cost equal to the maximum cost of all the columns in the problem. The perturbed columns are often selected for supplementary perturbations, and the result can be far from the initial solution. Note that the optimal solution columns are not removed from the problem. This method can obtain many instances (with many starting solutions) from an initial problem; more details can be found in Zaghrouti et al. (2014). The instances and their accompanying initial solutions are available for the OR community.

5.2 Numerical results and some implementation tips

The medium and large sets of instances are difficult to solve by CPLEX. Rosat et al. (2014a) mentioned in that, within a time limit of 10h, CPLEX could not find a feasible solution for the original problems of both sets. We present in Table 1 average results for the most recent version of CPLEX for which we provided initial integer solutions (Init.) that are exactly the same as the ones ISUD and ZOOM start from. Each entry in this table is an average of 30 instances. CPLEX is able to solve small instances to optimality in 7 s in average. These times are comparable to those of ZOOM. We set a time limit for the medium instances to 600 and to 3600 s for large instances. These time limits are at least 3 times far from the ZOOM times. The best integer solutions (Best int.) found by CPLEX are very far from the optimal solutions (Int. opt). Note that, for these reasons, we do not compare with CPLEX in the tables that follow.

Table 2 ZOOM versus ISUD (nonzeros)

Instances size	Orig. Pb.		RP (MIP)	
	NZs	NZs/Col.	NZs	NZs/Col.
Small	74,000	8	720	3
Medium	2,732,000	21	22,830	12
Large	10,942,000	19	314,232	11

The average size of the RPs solved as MIPs in Step 2 of ZOOM is small: Table 2 compares the average numbers (over 30 instances) of nonzero elements (total (NZs) and per column (NZs/col.)) of the original problem (Orig. Pb.) and the RPs (MIP) for the small, medium, and large instances. We would like to outline the fact that when we construct the RP in Step 2, we do not remove all redundant rows. We remove only rows that are identical in the same way of the paper (El Hallaoui et al. 2010) because this type of redundant rows is easy to identify. The number of nonzeros in the RP (MIP) is reduced by huge factors varying from 30 to more than 100 on average. The density is also reduced by a factor of 2. Therefore, we do not use the multiphase strategy when solving the RP in Step 2 of ZOOM; it is used only for CP.

Recall that the distance of a column to a given solution is its incompatibility degree, as explained in Sect. 2. We consider first the columns that are not too far from the best solution found and we increase this distance as needed. More precisely, we start by phase 1 when we solve CP_Q in Step 3 of ZOOM. As long as we do not find a descent direction, we increase by 1 the phase number, i.e., we go to the next phase and solve CP_Q again. The maximum phase number (maximum degree of incompatibility), used for the multiphase strategy, is set to 8. This value is reached in only one instance. We use the same stopping criteria (the best) for ISUD as in Zaghroui et al. (2014). We would like to mention that the results of ISUD are improved because the version of CPLEX has changed from 12.0 used in Zaghroui et al. (2014) to 12.4 here.

The first set of tables (Tables 3, 4, and 5) show from left to right: the unperturbed ratio for the instance (Orig. Col. %), the identifier for the instance (Instance #), the distance of the initial solution from the optimal one as a percentage of the optimal objective value (Instance Err. %), the distance to optimality of the solution found beside the total computational time in seconds (Objective Err. % beside Objective Time) for both ISUD and ZOOM, the total number of MIPs built and solved by ZOOM (Num), the number of MIPs that improved the current integer solution (Num+), the average number of rows in MIPs (Rows), the average number of columns in MIPs (Cols), the average number of nonzero elements in MIPs (NZs), and the average computational time in seconds for MIPs (Time).

The error percentages (Err. %) may be greater than 100 because they compare the objective value of a solution to the optimal objective value $[(\text{solution value} - \text{optimal value})/\text{optimal value}] \times 100$. The MIP values (rows, columns, NZs, and time) are average values aggregated over all the MIPs solved by ZOOM. When no MIP is called, no information is available.

The second set of tables (Tables 6, 7, and 8) compare the CPs for ISUD and ZOOM. The first two columns show the unperturbed ratio (Orig. Col. %) and the identifier for the instance (Instance #). Then, for both ISUD and ZOOM, the rest of columns show: the highest phase in which an integer solution is found (Phase), the total number of solutions, i.e., directions, found by CP (Total) that is also the number of iterations, the number of disjoint solutions (Disj.), the maximum size of the disjoint solutions (Max.) and the average size of the disjoint solutions (Avg.).

In both sets of tables, we have included average lines in bold to compare the average behavior of the two algorithms.

The results show that ZOOM outperforms ISUD in terms of solution quality, and it is generally slightly faster. Interestingly, ZOOM obtains solutions that are very close to optimality or optimal in all cases. We observe that ZOOM has the smallest error ratio average (almost 0% in average and a pic of 0.8% for ZOOM against 7.28% in average and a pic of 200% for ISUD). ZOOM is better than ISUD for all instances except for some small instances where ISUD is slightly and insignificantly better.

When we solve RP in Step 2 of ZOOM, we consider all columns that are compatible with ϱ , including those that are in phases higher than the actual phase of CP. To reach these columns, ISUD (its CP) has to go to similar high phases which complicates its execution: too much time for solving CP (too many columns to consider) and the directions found by CP are too much fractional because as we observed **non-integrality increases with the incompatibility degree, i.e., with the phase number**. In a highly fractional direction, the number of variables (v_j) taking nonzero values is huge actually. Consequently, ISUD stops with a poor solution due to its deep branching (setting to 0 these variables) as explained in Sect. 3. This is avoided in ZOOM.

Approximately 52% of the MIPs improve the current solution for difficult instances (see the sets with the unperturbed ratio equal to 20% in Tables 3, 4 and 5). When the MIP does not improve the current solution, we use this information as an indicator to increase the phase number and often we find an integer direction the next iteration. For some easy instances for which CP does not encounter a fractional solution, ZOOM does not build any MIPs and consequently it behaves the same way as ISUD.

ZOOM is in average slightly faster than ISUD, about 5% faster in average on difficult instances because ISUD is faster in some cases when it stops prematurely with a poor solution quality on 35% of the small and medium instances that are difficult (i.e., set with unperturbed ratio equal to 20%). Solving the MIPs is fast as expected (see discussion of Sect. 4): less than 3 s, on average, for the small and medium instances and around 9 s for the large instances. The number of columns considered is 20–30 times less than the total number of columns.

The results in Tables 6, 7, and 8 show that ZOOM explores less phase numbers than ISUD because some useful columns in higher phases are considered earlier in RP as mentioned above. Also, ZOOM solves fewer CPs and consequently does few iterations (8% less than ISUD on difficult instances). This can be explained by the fact that ZOOM uses also RP as an MIP to improve the current solution, interestingly by longer steps (ZOOM explores, in contrast to ISUD, some non adjacent solutions). For the same reason, the maximum and average sizes of the CP solutions are also smaller in ZOOM.

The lower bounds given by Proposition 2.9 are very good. For the most difficult (with regard to the phase we obtain the best solution, i.e., phase 8) instance (#87), $z_{\varrho}^{CP} = 0$ when we consider only columns with incompatibility degree not larger than 8 and $z_{\varrho}^{CP} = -7$ when we consider all columns. The cost of the (optimal) solution found is 3,302,462. From all above, we clearly see that the numerical results confirm the theoretical results presented in Sects. 2 and 4.

6 Possible extensions

The zooming approach is globally a primal exact approach. At each iteration, it improves the current integer solution by zooming around an improving fractional direction. In practice, it

Table 3 ZOOM versus ISUD (small instances)

Orig. Col. %	Instance		ISUD		ZOOM		MIP averages				NZs	Time
	#	Err. %	Objective Err. %	Time	Objective Err. %	Time	Num.	Num+	Rows	Cols		
50	1	360.6	0	7	0	7	4	0	127	198	667	0
	2	357.4	0	6	0	5	4	0	127	198	669	0
	3	347.7	0	6	0	5	4	0	127	201	684	0
	4	356.8	0.08	7	0	6	5	1	123	183	590	0
	5	353.3	0	6	0	5	4	0	127	200	676	0
	6	352.8	0	5	0	5	4	0	127	202	682	0
	7	364.2	0	6	0	5	4	0	127	198	669	0
	8	368	0	7	0.03	6	4	0	128	199	667	0
	9	360.1	0	6	0	5	4	1	130	217	751	0
	10	366.9	0	7	0	6	3	0	132	213	749	0
		358.78	0.01	6.3	0	5.5	4	0.2	127.5	200.9	680.4	0
35	11	472.1	0.03	6	0	5	4	1	128	196	664	0
	12	472.8	0	8	0.09	9	5	2	129	192	633	0.2
	13	464.7	0	5	0	5	4	0	127	200	675	0
	14	468.4	0	6	0	6	5	1	140	265	961	0
	15	463.5	0	6	0	5	3	0	132	214	753	0
	16	466.3	0	8	0.06	6	6	2	127	180	566	0
	17	456.3	0	11	0.8	11	7	4	42	139	563	0.01
	18	471.6	0	5	0	5	4	0	127	200	676	0
	19	469.2	0.64	8	0	9	4	1	125	189	626	0
	20	464.6	0	6	0	6	6	2	122	175	552	0
		466.95	0.07	6.9	0.1	6.7	4.8	1.3	119.9	195	666.9	0.02

Table 3 continued

Orig. Col. %	Instance		ISUD		ZOOM		MIP averages					
	#	Err. %	Objective		Objective		Num.	Num+	Rows	Cols	NZs	Time
			Err. %	Time	Err. %	Time						
20	21	570.2	0	8	0	8	2	0	140	248	933	0
	22	561	0	12	0	11	5	2	173	438	1778	0.2
	23	559.9	138.19	9	0.05	9	5	2	120	165	519	0
	24	557.7	0	9	0	9	3	0	132	214	754	0
	25	562.5	0	12	0.25	14	9	6	128	212	692	0
	26	561	0	9	0	9	4	1	127	192	646	0
	27	573.1	0	8	0.05	8	4	0	126	191	637	0
	28	569.6	0.02	10	0	9	5	3	126	185	614	0
	29	569.3	0	12	0	10	4	0	126	188	624	0
	30	573.7	200.63	8	0.09	13	13	10	126	199	615	0
		565.8	33.88	9.7	0.04	10	5.4	2.4	132.4	223.2	781.2	0.02

Averages (for the ten entries above) are in bold

Table 4 ZOOM versus ISUD (medium instances)

Orig. Col. %	Instance		ISUD		ZOOM		MIP averages			Cols	NZs	Time	
	#	Err. %	Objective	Time	Objective	Err. %	Time	Num.	Num+				Rows
50	31	53.43	0	40	0	0	41	1	0	100	153	1021	0
	32	51.39	0	57	0	0	58	1	0	100	153	1021	0
	33	53.43	0	56	0	0	55	1	0	100	153	1021	0
	34	53.44	0	55	0	0	54	1	0	100	153	1021	0
	35	51.38	0	91	0	0	85	3	1	207	1103	13192	0
	36	51.38	0	41	0	0	41	1	0	100	153	1021	0
	37	51.38	0	104	0	0	97	2	0	239	1367	16,629	0
	38	51.38	0	98	0	0	93	2	0	239	1370	16,664	0
	39	51.38	0	124	0	0	104	3	0	297	2084	26,049	0
	40	51.38	0	39	0	0	38	1	0	100	153	1021	0
		52.00	0	70.5	0	0	66.6	1.6	0.1	158.2	684.2	7866	0
35	41	65.77	0	136	0	0	127	3	1	189	942	11,189	0
	42	65.77	0	63	0	0	63	1	0	100	153	1021	0
	43	65.77	10.27	92	0	0	101	5	2	196	1043	12,418	0.2
	44	67.83	12.33	130	0	0	139	4	2	363	4667	64,402	2.75
	45	65.77	0	180	0	0	147	6	4	207	2067	28,428	1.78
	46	67.83	0	105	0	0	101	2	0	238	1361	16,443	0
	47	67.82	0	193	0	0	125	3	1	310	3248	41,342	0.67
	48	65.77	0	75	0	0	70	2	0	239	1369	16,645	0
	49	65.77	0	107	0	0	90	3	1	211	1094	13,016	0
	50	67.83	0	114	0	0	98	2	0	239	1370	16,656	0.5
		66.59	2.26	119.5	0	0	106.1	3.1	1.1	229.2	1731.4	22,156	0.59

Table 4 continued

Orig. Col. %	Instance		ISUD		ZOOM		MIP averages					
	#	Err. %	Objective		Objective		Num.	Num+	Rows	Cols	NZs	Time
			Err. %	Time	Err. %	Time						
20	51	82.21	0	213	0	109	3	1	439	5162	65,688	1
	52	82.21	0	146	0	141	2	0	238	1360	16,445	0.5
	53	82.21	0	78	0	77	1	0	100	153	1021	0
	54	82.21	0	133	0	123	2	0	239	1371	16,658	0
	55	82.21	41.11	75	0	106	2	1	497	9228	120,243	6
	56	82.21	63.72	38	0	86	2	1	552	138,69	194,241	13
	57	82.21	0	222	0	149	3	1	370	3110	40,936	0.67
	58	82.21	22.61	111	0	170	4	1	359	4660	64,203	3.25
	59	82.21	0	101	0	95	2	0	239	1369	16,641	0
	60	82.21	32.88	63	0	81	2	1	455	7007	94,194	2
		82.21	16.03	118.0	0	113.7	2.3	0.6	348.8	4728.9	63,027	2.64

Averages (for the ten entries above) are in bold

Table 5 ZOOM versus ISUD (large instances)

Orig. Col. %	Instance		ISUD		ZOOM		MIP averages				NZs	Cols	Rssows	Time			
	#	Err. %	Objective	Time	Objective	Time	Num.	Num+	Num.	Time							
50	61	51.88	0	652	0	655	0	0	0	0	0	0	0	0	0	0	0
	62	50.34	0	919	0	912	0	0	0	0	0	0	0	0	0	0	0
	63	50.34	0	307	0	307	0	0	0	0	0	0	0	0	0	0	0
	64	50.35	10.68	789	0	916	1	1	1	716	35,387	377,443	9				
	65	50.34	0	1355	0	1364	0	0	0	0	0	0	0	0	0	0	0
	66	51.87	0	274	0	273	0	0	0	0	0	0	0	0	0	0	0
	67	50.34	10.68	453	0	1141	1	1	1	681	27,519	304,478	7				
	68	50.35	0	623	0	628	0	0	0	0	0	0	0	0	0	0	0
	69	51.87	0	587	0	587	0	0	0	0	0	0	0	0	0	0	0
	70	50.34	0	1338	0	1342	0	0	0	0	0	0	0	0	0	0	0
		50.80	2.14	729.7	0	812.5	0.2	0.2	0.2	698.5	31453	340960	8				
35	71	65.60	0	804	0	797	0	0	0	0	0	0	0	0	0	0	0
	72	65.60	0	695	0	695	0	0	0	0	0	0	0	0	0	0	0
	73	65.59	0	617	0	611	0	0	0	0	0	0	0	0	0	0	0
	74	65.60	0	1266	0	1257	0	0	0	0	0	0	0	0	0	0	0
	75	65.60	15.26	352	0	664	1	1	1	727	30,366	340,469	8				
	76	65.60	0	1508	0	1501	0	0	0	0	0	0	0	0	0	0	0
	77	65.60	19.83	251	0	350	1	1	1	755	31,874	352,756	10				
	78	65.60	0	1929	0	1938	0	0	0	0	0	0	0	0	0	0	0
	79	65.59	0	1578	0	1447	1	1	1	673	22,763	240,986	6				
	80	65.60	10.68	416	0	681	1	1	1	651	25,743	272,085	7				
		65.60	4.58	941.6	0	994.4	0.4	0.4	0.4	701.5	27686.5	301574	7.75				

Table 5 continued

Orig. Col. %	Instance		ISUD		ZOOM		MIP averages					
	#	Err. %	Objective	Time	Objective	Time	Num.	Num+	Rssows	Cols	NZs	Time
		Err. %										
20	81	82.38	0	2126	0	2128	0	0	761	31,931	347,784	8
	82	80.85	0	1498	0	1219	1	1	0	0	0	0
	83	80.85	0	1473	0	1475	0	0	717	35,626	399,536	9
	84	80.86	0	1695	0	1393	1	1	686	29,838	317,317	7
	85	80.86	0	1719	0	1578	1	1	644	21,032	222,389	19
	86	80.86	0	1031	0	1010	0	0	741	40,274	453,855	11
	87	80.85	0	1675	0	1390	2	1	596	15,818	164,729	4
	88	80.85	0	2018	0	2010	0	0	690.83	29086.5	317601.7	9.67
	89	80.86	0	1299	0	1174	1	1				
	90	80.86	0	1666	0	881	1	1				
		81.01	0	1620	0	1425.8	0.7	0.6				

Averages (for the ten entries above) are in bold

Table 6 ZOOM CP versus ISUD CP (small instances)

Orig. Col. %	#	ISUD					ZOOM				
		CP solutions					CP solutions				
		Phase	Total	Disj.	Max.	Avg.	Phase	Total	Disj.	Max.	Avg.
50	1	4	35	29	6	2.4	4	33	29	6	2.4
	2	3	38	31	7	2.4	3	36	31	7	2.4
	3	4	33	27	4	2.4	4	31	27	4	2.4
	4	5	56	39	8	2.9	3	42	36	7	2.7
	5	4	33	27	4	2.4	4	31	27	4	2.4
	6	4	33	27	5	2.6	4	31	27	5	2.6
	7	4	38	32	5	2.2	4	36	32	5	2.2
	8	4	41	33	9	2.9	4	36	31	9	2.8
	9	5	32	26	7	2.5	4	30	25	6	2.5
	10	4	36	31	10	2.5	4	34	31	10	2.5
	4.1	37.5	30.2	6.5	2.52	3.8	34	29.6	6.3	2.49	
35	11	5	42	33	7	3	4	35	31	11	3
	12	6	39	29	19	3.4	4	40	34	8	3
	13	3	37	31	7	2.6	3	35	31	7	2.6
	14	5	35	27	17	3.1	4	34	29	6	2.6
	15	4	37	32	5	2.8	4	35	32	5	2.8
	16	5	50	39	8	2.8	4	39	33	7	2.6
	17	5	43	36	10	3.5	5	45	37	7	3.5
	18	3	46	39	5	2.3	3	44	39	5	2.3
	19	4	46	37	10	3.1	4	42	38	10	3.1
	20	5	40	30	7	2.7	6	22	20	3	2.2
	4.5	41.5	33.3	9.5	2.93	4.1	37.1	32.4	6.9	2.77	
20	21	5	44	40	15	3.3	5	42	40	15	3.3
	22	5	53	44	8	3.1	5	46	41	8	3.2
	23	5	44	23	18	4.3	5	31	26	14	4.5
	24	4	48	43	13	3.3	4	46	43	13	3.3
	25	5	52	43	13	3.6	5	52	42	9	3.2
	26	5	42	33	12	3.4	4	36	32	12	3.4
	27	4	47	40	8	2.8	4	43	39	8	2.8
	28	6	51	40	8	3.2	5	45	40	7	3
	29	4	63	51	14	3.3	4	47	41	14	3.4
	30	4	43	31	11	3.5	5	47	34	17	3.8
	4.7	48.7	38.8	12	3.38	4.6	43.5	37.8	11.7	3.39	

Averages (for the ten entries above) are in bold

improves on the ISUD algorithm (Zaghroui et al. 2014). Possible extensions of this work include:

- Considering non-SPP constraints: non-SPP constraints are added to many SPPs to model various regulations. For example, in aircrew scheduling there may be a constraint limiting the number of flying hours per base. These constraints represent less than 10% of the

Table 7 ZOOM CP versus ISUD CP (medium instances)

Orig. Col. %	#	ISUD					ZOOM				
		CP solutions					CP solutions				
		Phase	Total	Disj.	Max.	Avg.	Phase	Total	Disj.	Max.	Avg.
50	31	4	15	14	3	2.1	4	15	14	3	2.1
	32	5	17	16	3	2.1	5	17	16	3	2.1
	33	5	16	15	3	2.2	5	16	15	3	2.2
	34	5	15	14	4	2.4	5	15	14	4	2.4
	35	6	15	12	6	2.4	6	15	12	3	2.2
	36	5	16	15	3	2.1	5	16	15	3	2.1
	37	6	18	16	3	2.1	6	18	16	3	2.1
	38	6	18	16	3	2.1	6	18	16	3	2.1
	39	7	19	15	5	2.3	7	18	15	5	2.3
	40	4	15	14	3	2.1	4	15	14	3	2.1
		5.3	16.4	14.7	3.6	2.19	5.3	16.3	14.7	3.3	2.17
35	41	0.6	24	21	4	2.3	6	23	20	3	2.2
	42	5	18	17	4	2.3	5	18	17	4	2.3
	43	6	19	15	3	2.3	6	20	15	3	2.3
	44	7	22	17	3	2.2	7	20	16	3	2.2
	45	7	23	19	5	2.5	6	21	15	5	2.5
	46	6	22	20	6	2.3	6	22	20	6	2.3
	47	7	24	20	4	2.4	6	20	17	4	2.4
	48	6	16	14	8	2.7	6	16	14	8	2.7
	49	6	22	19	5	2.3	6	20	17	5	2.4
	50	6	22	20	3	2.2	6	22	20	3	2.2
		6.2	21.2	18.2	4.5	2.35	6	20.2	17.1	4.4	2.35
20	51	8	22	17	8	3.2	7	16	13	5	2.5
	52	6	26	24	4	2.6	6	26	24	4	2.6
	53	5	25	24	5	2.5	5	25	24	5	2.5
	54	6	25	23	5	2.7	6	25	23	5	2.7
	55	6	16	15	4	2.5	6	17	15	4	2.5
	56	6	9	8	3	2.1	6	10	8	3	2.1
	57	8	27	21	5	2.6	7	23	20	5	2.5
	58	6	21	18	4	2.6	7	23	19	4	2.5
	59	6	23	21	5	2.6	6	23	21	5	2.6
	60	6	16	15	5	2.4	6	17	15	5	2.4
		6.3	21.0	18.6	4.8	2.58	6.2	20.5	18.2	4.5	2.49

Averages (for the ten entries above) are in bold

total number of constraints, and the problem becomes an SPP with side constraints. The zooming algorithm can easily handle these additional constraints by putting them in the RP. Other options could be studied.

- Accelerating the solution process by obtaining several orthogonal descent directions, good in practice for large problems, by successively solving appropriate CPs. To do

Table 8 ZOOM CP versus ISUD CP (large instances)

Orig. Col. %	#	ISUD					ZOOM				
		CP solutions					CP solutions				
		Phase	Total	Disj.	Max.	Avg.	Phase	Total	Disj.	Max.	Avg.
50	61	6	18	18	4	2.4	6	18	18	4	2.4
	62	6	19	19	4	2.2	6	19	19	4	2.2
	63	5	17	17	4	2.2	5	17	17	4	2.2
	64	8	23	16	3	2.1	5	16	15	2	2
	65	7	16	16	4	2.5	7	16	16	4	2.5
	66	5	19	19	3	2.1	5	19	19	3	2.1
	67	7	16	13	4	2.3	7	14	13	4	2.3
	68	6	17	17	4	2.2	6	17	17	4	2.2
	69	6	16	16	4	2.4	6	16	16	4	2.4
	70	7	19	19	3	2.2	7	19	19	3	2.2
		6.3	18	17.1	3.7	2.25	6	17.1	16.9	3.6	2.24
35%	71	6	25	25	3	2.2	6	25	25	3	2.2
	72	6	23	23	5	2.5	6	23	23	5	2.5
	73	6	24	24	5	2.3	6	24	24	5	2.3
	74	7	26	26	5	2.3	7	26	26	5	2.4
	75	6	23	21	5	2.3	6	22	21	5	2.3
	76	7	24	24	5	2.4	7	24	24	5	2.4
	77	5	18	17	5	2.5	5	18	17	5	2.5
	78	5	24	24	5	2.3	7	26	26	5	2.3
	79	8	27	25	4	2.6	7	24	23	4	2.6
	80	6	23	22	7	2.5	6	23	22	7	2.6
		6.2	23.7	23.1	4.9	2.39	6.3	23.5	23.1	4.9	2.4
20%	81	7	27	27	7	3.4	7	27	27	7	3.4
	82	8	24	22	13	3.7	7	19	18	6	3
	83	7	31	30	7	2.5	7	31	30	7	2.5
	84	8	23	21	13	3.4	7	18	17	4	2.4
	85	8	30	27	8	3.3	7	26	25	8	3.2
	86	6	32	32	6	2.6	6	32	32	6	2.6
	87	9	25	21	22	3.8	8	20	18	22	3.7
	88	7	31	31	8	3.2	7	31	31	8	3.2
	89	8	21	19	10	3.6	7	17	16	6	2.9
	90	7	34	33	4	2.6	6	32	31	4	2.6
		7.5	27.8	26.3	9.8	3.21	6.9	25.3	24.5	7.8	2.96

Averages (for the ten entries above) are in bold

this, we solve the CP, remove the variables with positive values in the solution and the constraints that they cover, and start again. This gives a multidirectional zoom on a targeted neighborhood around the improving directions. The number of directions should be customizable and adjusted experimentally; in this paper we consider just one direction.

- Obtaining good improving directions by adjusting the coefficients of the normalization constraint or by cutting bad directions: Rosat et al. developed the concepts of penalizing and cutting fractional directions in Rosat et al. (2014a, b). These concepts could be adapted for the zooming approach.
- Adapt the ZOOM algorithm for other kind of SPP problems. The actual partial pricing based on the incompatibility degree is designed for routing or scheduling problems. A more smarter partial pricing strategy should be ideally a function of the reduced cost, the incompatibility degree, the number of nonzeros (nonzero elements), and other relevant attributes of the problem.
- Integrating with column generation: in practice, large SPPs are often solved by column generation. We could explore how to use this column generation technique in the vector space decomposition framework.

References

- Balas, E., & Padberg, M. W. (1972). On the set-covering problem. *Operations Research*, *20*, 1152–1161.
- Balas, E., & Padberg, M. W. (1975). On the set-covering problem: II. An algorithm for set partitioning. *Operations Research*, *23*, 74–90.
- Balas, E., & Padberg, M. W. (1976). Set partitioning: A survey. *SIAM Review*, *18*, 710–760.
- Balinski, M. L., & Quandt, R. E. (1964). On an integer program for a delivery problem. *Operations Research*, *12*, 300–304.
- Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., & Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, *46*, 316–329.
- Bouarab, H., El Hallaoui, I., Metrane, A., & Soumis, F. (2017). Dynamic constraint and variable aggregation in column generation. *European Journal of Operational Research*, <https://doi.org/10.1016/j.ejor.2017.04.049>.
- Christofides, N., & Paixao, J. (1993). Algorithms for large scale set covering problems. *Annals of Operations Research*, *43*, 259. <https://doi.org/10.1007/BF02025297>.
- Cordeau, J.-F., Desaulniers, G., Lingaya, N., Soumis, F., & Desrosiers, J. (2001). Simultaneous locomotive and car assignment at VIA Rail Canada. *Transportation Research B*, *35*, 767–787.
- Cullen, F., Jarvis, J., & Ratliff, H. (1981). Set partitioning based heuristics for interactive routing. *Networks*, *11*, 125–143.
- Desrochers, M., Desrosiers, J., & Soumis, F. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, *40*, 342–354.
- Desrochers, M., & Soumis, F. (1989). A column generation approach to the urban transit crew scheduling problem. *Transportation Science*, *23*, 1–13.
- El Hallaoui, I., Metrane, A., Desaulniers, G., & Soumis, F. (2011). An improved primal simplex algorithm for degenerate linear programs. *INFORMS Journal on Computing*, *23*(4), 569–577.
- El Hallaoui, I., Metrane, A., Soumis, F., & Desaulniers, G. (2010). Multi-phase dynamic constraint aggregation for set partitioning type problems. *Mathematical Programming A*, *123*(2), 345–370.
- Gamache, M., Soumis, F., Marquis, G., & Desrosiers, J. (1999). A column generation approach for large scale aircrew rostering problems. *Operations Research*, *47*, 247–263.
- Gomory, R. E. (1958). Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, *64*(5), 275–278.
- Gomory, R. E. (1963). An all-integer integer programming algorithm. In F. M. John., & L. T. Gerald (Eds.), *Industrial scheduling* (pp. 193–206). Englewood Cliffs, NJ: Prentice Hall.
- Gopalakrishnan, B., & Johnson, E. L. (2005). Airline crew scheduling: State-of-the-art. *Annals of Operations Research*, *140*, 305. <https://doi.org/10.1007/s10479-005-3975-3>.
- Haase, K., Desaulniers, G., & Desrosiers, J. (2003). Simultaneous vehicle and crew scheduling in urban mass transit systems. *Transportation Science*, *35*(3), 286–303.
- Haus, U., Köppe, M., & Weismantel, R. (2001). The integral basis method for integer programming. *Mathematical Methods of Operations Research*, *53*(3), 353–361.
- Hoffman, K. L., & Padberg, M. (1993). Solving airline crew scheduling problems by branch-and-cut. *Management Science*, *39*, 657–682.

- Iqbal Ali, A. (1998). Reformulation of the set partitioning problem as a pure network with special order set constraints. *Annals of Operations Research*, 81, 233. <https://doi.org/10.1023/A:1018953006796>.
- Letchford, A. N., & Lodi, A. (2002). Primal cutting plane algorithms revisited. *Mathematical Methods of Operations Research*, 56(1), 670–81.
- Lubbecke, M. E., & Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, 53(6), 1007–1023.
- Rönnberg, E., & Torbjörn, L. (2009). Column generation in the integral simplex method. *European Journal of Operational Research*, 192, 333–342.
- Rosat, S., El Halaoui, I., & Soumis, F. (2014a). Influence of the normalization constraint on the integral simplex using decomposition. *Discrete Applied Maths.*, <https://doi.org/10.1016/j.dam.2015.12.015>.
- Rosat, S., El Halaoui, I., Soumis, F., & Lodi, A. (2014b). Integral simplex using decomposition with primal cuts. Lecture Notes in Computer Science (Springer), SEA 2014 (Symposium on Experimental Algorithms) Proceedings.
- Rosat, S., El Halaoui, I., Soumis, F., & Lodi, A. (2017). Integral simplex using decomposition with primal cuts. *Mathematical Programming.*, <https://doi.org/10.1007/s10107-017-1123-x>.
- Thompson, G. L. (2002). An integral simplex algorithm for solving combinatorial optimization problems. *Computational Optimization and Applications*, 22, 351–367.
- Zaghrouti, A., Soumis, F., & El Hallaoui, I. (2014). Integral simplex using decomposition for the set partitioning problem. *Operations Research*, 62, 435–449. <https://doi.org/10.1287/opre.2013.1247>.