Les Cahiers du GERAD ISSN: 0711–2440

Dynamic constraint and variable aggregation in column generation

H. Bouarab, I. El Hallaoui,
A. Metrane, F. Soumis

G–2014–82

November 2014
Revised: November 2015

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

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

Dynamic constraint and variable aggregation in column generation

Hocine Bouarab
Issmail El Hallaoui
Abdelmoutalib Metrane
François Soumis

GERAD & Département de mathématiques et de génie industriel, Polytechnique Montréal, Montréal (Québec) Canada, H3C 3A7

hocine.bouarab@polymtl.ca
issmail.elhallaoui@polymtl.ca
abdelmoutalib.metrane@gerad.ca
francois.soumis@gerad.ca

November 2014
Revised: November 2015

Les Cahiers du GERAD
G–2014–82

Abstract: Starting from the improved primal simplex (IPS) decomposition, introduced by Elhallaoui et al. [7] to tackle degeneracy in general linear programs, we introduce and discuss the mathematical foundations of improved column-generation decompositions (ICG) that are better than the standard column-generation decomposition when degeneracy is an issue. We also present an improved dynamic constraint aggregation (IDCA), which is a specialization of ICG to efficiently solve set partitioning problems. We show that IDCA improves the dynamic constraint aggregation (DCA) and the multi-phase dynamic constraint aggregation (MPDCA) algorithms [5, 6] that not only reduce degeneracy but also profit from it to efficiently solve set partitioning problems. IDCA solves at each iteration a complementary problem (CP) to obtain a group of variables that can be aggregated or pivoted into the basis to bypass degenerate pivots and decrease the objective value and to obtain more "central" dual solutions to generate new columns. Our numerical results show reduction factors in the solution times exceeding 10 for IDCA compared to a state-of-the-art standard column-generation solver.

Key Words: Column generation, degeneracy, dynamic constraint aggregation, improved primal simplex, set partitioning problems.

1 Introduction

Column generation (CGen) is widely used to solve many industrial problems, especially in transportation. It is generally embedded in sophisticated solution methodologies such as branch-and-price (branch-and-bound where the lower bounds are computed by CGen; see [1]) that are able to solve these huge integer linear programs. In CGen, we solve (to optimality) at each iteration a linear relaxation of the so-called restricted master problem (RMP). We thus obtain a primal and a dual solution. The dual solution is used to price out columns (variables) in one or many subproblems. New columns with negative reduced costs (in a minimization problem) are appended to the RMP, and we iterate until no columns with negative reduced costs are identified or the stopping criteria are met. More details on CGen can be found in [12].

If the RMP is highly degenerate, the standard CGen method becomes inefficient. Degeneracy slows down the CGen by increasing the number of iterations. This is because the primal simplex algorithm, generally used to solve the RMP, selects almost randomly one of the many dual solutions associated with a degenerate primal solution to price out columns in the subproblem. These dual solutions oscillate and the generated columns are of poor quality.

In many vehicle and crew scheduling problems the subproblem is generally a shortest path problem with resource constraints that we solve by dynamic programming (see [2, 3, 8]). Solving just one such subproblem may be costly, especially if the number of resource constraints exceeds four or five. In some aircrew applications, we have to solve hundreds of subproblems with more than 10 resource constraints. In this situation, reducing the number of iterations can have a large impact.

Many vehicle and crew scheduling problems are modeled as set partitioning problems (SPPs), and these are known to be highly degenerate. These problems cover a set of tasks (flight legs, bus trips, or customers to visit) exactly once with feasible paths (vehicle routes or crew schedules) at a minimum cost. Large instances may involve more than one billion variables and several thousand constraints. More than 90% of the constraints are set partitioning constraints.

A promising method to handle degeneracy in set partitioning problems was proposed in 2005 by Elhallaoui et al. [5]. This method, called dynamic constraint aggregation (DCA), aggregates some of the set partitioning constraints; it reduces degeneracy and the pivoting time by reducing the basis size. Moreover, variables that are compatible with the aggregation are favored as entering variables, to produce nondegenerate pivots and to keep the basis size as small as possible. In fact, a compatible variable can be pivoted into the basis of the aggregated problem without modifying the aggregation.

There are many dual solutions for each degenerate primal solution. At each iteration, DCA selects a good dual solution via a heuristic procedure that solves a shortest path problem. This procedure aims to find dual values for which incompatible variables with some special structure have a nonnegative reduced cost.

Significant improvements to DCA are presented in [6], which presents a measure of the degree of incompatibility of the variables. Based on this measure, the authors propose a multiphase partial pricing strategy (called multiphase dynamic constraint aggregation or MPDCA) that prices out, at the beginning of the solution process, only the variables that are compatible with or just slightly incompatible with the current aggregation. As the solution process progresses, variables with a greater degree of incompatibility are priced out, if necessary, to ensure the exactness of the overall solution. This strategy avoids a rapid increase in the basis size because we can reach an optimal (final) solution by considering, at each iteration, slightly incompatible variables. MPDCA uses the heuristic procedure of DCA to find dual values for the discarded rows. The dual values found by the heuristic could be unstable if too few incompatible variables are considered or they are not sufficiently representative; this could be considered the main weakness of DCA and MPDCA.

Elhallaoui et al. [7] propose an improved primal simplex algorithm (IPS) for general linear programs that is much more efficient in the presence of degeneracy than the traditional primal simplex method. As in [6], the problem is reduced by removing from the basis the degenerate variables and constraints. The dual variables of the discarded rows are computed by solving a linear problem, called the complementary problem (CP). When there is no degeneracy, this algorithm coincides with the traditional primal simplex algorithm. When degeneracy occurs, the method takes advantage of it by solving to optimality a reduced problem (RP)

that involves only compatible variables. The CP finds descent directions, combining or aggregating many variables, in the complementary subspace of incompatible variables, if the solution is not optimal yet. In IPS the variables are assumed to be known a priori, i.e., there is no CGen.

In this paper we present interesting theoretical and numerical contributions. We develop an improved column generation method (ICG) based on three-level decompositions. Traditionally, there are two levels in standard CGen: the RMP and the subproblem(s). If we want to improve the dual values, we must act directly on the RMP. We could perturb its primal feasibility; for example, stabilization techniques add stabilization variables. Alternatively, we could use interior point methods, but these methods often increase the computational time and yield a highly fractional solution that negatively impacts the branch-and-price process. In this paper, we introduce three-level CGen decompositions where the added intermediate level is CP. The role of this supplementary level is twofold. First, it is used to select good columns from those produced by the subproblem to add to the reduced RMP. Second, it efficiently derives stable dual values for all the constraints from an aggregated dual vector corresponding to the RP constraints. The CP is easier to solve and to handle than the RMP, giving better dual values for a lower computational cost. This supplementary level adds flexibility to the CGen approach. We also use the CP to aggregate incompatible variables (columns). These columns are used to improve the dual solutions of the RP without increasing its basis size.

Actually, there are many ways of extending IPS to the general CGen context. A first way (based on only one three-level decomposition) is given in [4]. The authors discussed basic methodological aspects of this extension, i.e. the optimality condition and the construction of the subproblems. They do not explore its efficiency from theorectical point of view, nor they provide any numerical results. We will study some strengths and weaknesses of this extension and present two more efficient ways of integerating IPS and CGen.

We further propose an improved dynamic constraint aggregation (IDCA) for SPPs that combines the strengths of DCA, MPDCA, and IPS. We show that IDCA generalizes and significantly improves on DCA and MPDCA. IDCA is highly stable compared to the previous methods. We present new theoretical results on the quality of the dual solutions. Our numerical results show that the new method is 10 times faster than a state-of-the-art commercial CGen solver.

This paper is organized as follows. In Section 2, we present the IPS decomposition. General properties of compatibility matrices and partial pricing strategies are presented in this section. In Section 3, we present the ICG decompositions that combine IPS and standard CGen decompositions. Section 4 describes the IDCA, a specialization of ICG to SPPs, and discusses a specialized compatibility matrix for some subtypes of SPPs. In Section 5, we present numerical results for instances of vehicle and crew scheduling problems. Section 6 provides concluding remarks.

2 IPS decomposition

In this section, we present the IPS decomposition for degenerate linear programs, discuss compatibility matrices, and derive partial pricing strategies.

2.1 From degeneracy to decomposition

Let LP be a linear program defined as follows:

$$(LP) \qquad z^{\text{LP}} = \min_{x} \quad c \cdot x \tag{2.1}$$

$$\text{s.t.} \quad Ax = b \tag{2.2}$$

$$x \geq 0 \tag{2.3}$$

where $x \in \mathbb{R}^n$, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$ (a_{ij} is an entry of A), and x is the vector of variables. We refer to a column of A as A_j, $j \in J = \{1, \ldots, n\}$.

Let x be a basic solution to LP. Note that x is often degenerate, especially in the case of the SPP. Let $\mathcal{s}$ be an index set of linearly independent columns containing at least the columns A_j with positive x_j; clearly $|\mathcal{s}| \leq m$. The set $\mathcal{s}$ is callled nondegenerate if it contains only positive variables indices and degenerate otherwise. Let $A_{\mathcal{S}} = \begin{pmatrix} A^1_{\mathcal{S}} \\ A^2_{\mathcal{S}} \end{pmatrix}$ be a submatrix of A composed of columns indexed in $\mathcal{s}$ where $A^1_{\mathcal{S}}$ is without loss of generality composed of the first —$\mathcal{s}$— linearly independent rows. $A^2_{\mathcal{S}}$ is of course composed of dependent rows.

Definition 2.1 *A column or a convex combination of columns is said to be compatible with $\mathcal{s}$ if it is linearly dependent on the columns of $\mathcal{s}$.*

The index set of compatible columns is denoted $\mathcal{c}_{\mathcal{S}}$ and the index set of incompatible columns is denoted $\mathcal{I}_{\mathcal{S}}$. Note that $\mathcal{s} \subseteq \mathcal{c}_{\mathcal{S}}$ because each column is linearly dependent of itself. When we restrict LP to the compatible columns $\mathcal{c}_{\mathcal{S}}$ and, therefore, to the first —$\mathcal{s}$— linearly independent rows, we obtain what we call the RP, which is formulated as:

$$(RP_{\mathcal{S}}) \quad \min_{x_{\mathcal{C}_{\mathcal{S}}}} \quad c^{\top}_{\mathcal{C}_{\mathcal{S}}} x_{\mathcal{C}_{\mathcal{S}}} \tag{2.4}$$

$$\text{s.t.} \quad A^1_{\mathcal{C}_{\mathcal{S}}} x_{\mathcal{C}_{\mathcal{S}}} = b_{\mathcal{S}} \tag{2.5}$$

$$x_{\mathcal{C}_{\mathcal{S}}} \geq 0 \tag{2.6}$$

where $c_{\mathcal{C}_{\mathcal{S}}}$ is the subvector of the costs of the compatible variables, $x_{\mathcal{C}_{\mathcal{S}}}$ is the subvector of the compatible variables, $A^1_{\mathcal{C}_{\mathcal{S}}}$ is the submatrix restricted to the compatible columns and the first $|\mathcal{s}|$ linearly independent rows, and $b_{\mathcal{S}}$ is the subvector of the right-hand side restricted to the first $|\mathcal{s}|$ components of b.

We solve RP to optimality using the simplex algorithm. Intuitively, RP is nondegenerate or slightly degenerate if $\mathcal{s}$ is nondegenerate or slightly degenerate. Obviously, the higher the degeneracy of the original problem, the smaller the RP. Consequently, pivots in RP are faster and less degenerate than in the original problem.

Pivoting on individual incompatible variables, i.e. variables that are not present in RP, does not improve the solution of RP and yields degeneracy. Hence, only incompatible columns forming a compatible convex combination are priced out. A CP is used to build such combinations and can be formulated as follows:

$$(CP'_{\mathcal{S}}) \qquad z^{CP'}_{\mathcal{S}} = \sum_{j \in \mathcal{I}_{\mathcal{S}}} v_j c_j - \sum_{\ell \in \mathcal{S}} \lambda_\ell c_\ell \tag{2.7}$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{I}_{\mathcal{S}}} v_j A_j = \sum_{\ell \in \mathcal{S}} \lambda_\ell A_\ell \tag{2.8}$$

$$e \cdot v = 1 \tag{2.9}$$

$$v \geq 0. \tag{2.10}$$

.

The objective function (2.7) is the reduced cost and measures the "improvement" in the objective value we obtain after entering the incompatible variables with $v_j > 0$. Constraint (2.8), where ($lambda_\ell$) are real numbers, ensure the compatibility of the selected combination ($\sum_{j \in \mathcal{I}_{\mathcal{S}}} v_j A_j$) according to definition 2.1. The constraint (2.9) is a normalization constraint (CP is unbounded without this constraint); $e = (1, \cdots, 1)^{\top}$ is of dimension dictated by the context. The real variables λ can be eliminated to obtain an equivalent model

$CP'_{\mathcal{S}}$ by using the fact that the columns A_ℓ are linearly independent.

$$(CP_{\mathcal{S}}) \qquad z^{CP}_{\mathcal{S}} = \min_{v} \quad \left(c^{\top}_{\mathcal{I}_{\mathcal{S}}} - c^{\top}_{\mathcal{S}} \left(A^1_{\mathcal{S}}\right)^{-1} A^1_{\mathcal{I}_{\mathcal{S}}}\right) v \tag{2.11}$$

$$\text{s.t.} \quad \left(A^2_{\mathcal{S}} \left(A^1_{\mathcal{S}}\right)^{-1} A^1_{\mathcal{I}_{\mathcal{S}}} - A^2_{\mathcal{I}_{\mathcal{S}}}\right) v = 0 \tag{2.12}$$

$$e \cdot v = 1 \tag{2.13}$$

$$v \geq 0. \tag{2.14}$$

where $\begin{pmatrix} A^1_{\mathcal{I}\mathcal{S}} \\ A^2_{\mathcal{I}_{\mathcal{S}}} \end{pmatrix} = (a_{ij})_{\substack{1 \leq i \leq m \\ j \in \mathcal{I}_{\mathcal{S}}}}$ with $A^1_{\mathcal{I}_{\mathcal{S}}}$ a $|\mathcal{S}| \times |\mathcal{I}_{\mathcal{S}}|$ matrix, and $v \in \mathbb{R}^{|\mathcal{I}_{\mathcal{S}}|}$.

The optimal solution to $CP_{\mathcal{S}}$ is a minimal convex combination in the sense that no convex combination of a strict subset of its *support* columns (i.e. corresponding to variables with $v_j > 0$) is compatible with $\mathcal{S}$ (proof is in [7]). We note that the columns of this optimal convex combination have the same reduced cost; this can be seen clearly from the dual of $CP'_{\mathcal{S}}$, denoted $CD_{\mathcal{S}}$. The models $CD_{\mathcal{S}}$, $CP'_{\mathcal{S}}$ and $CP_{\mathcal{S}}$ have the same optimal value.

$$(CD_{\mathcal{S}}) \qquad z^{CD}_{\mathcal{S}} = \max_{y,\, \alpha} \quad y \tag{2.15}$$

$$\text{s.t.} \quad c_j - \alpha \cdot A_j = 0, \qquad \forall j \in \mathcal{S} \tag{2.16}$$

$$c_j - \alpha \cdot A_j \geq y, \qquad \forall j \in \mathcal{I}_{\mathcal{S}} \tag{2.17}$$

where α is the dual vector corresponding to the constraints of the original problem.

We present a pseudocode for IPS in Algorithm 1. Below we make some interesting comments on IPS:

Remark 2.2

- *Algorithm 1 is optimal, i.e., it finds an optimal solution.*
- *When $\mathcal{S}$ is nondegenerate, the objective value decreases at each iteration of Algorithm 1.*
- *If we consider only a strict subset of $\mathcal{I}_{\mathcal{S}}$ in Algorithm 1, then the objective function value decreases at each iteration but we cannot confirm optimality if $z^{CP}_{\mathcal{S}} \geq 0$*

Algorithm 1 Improved Primal Simplex

Step 0. Find an initial solution indexed by $\mathcal{S}$.
The initial solution is not required to be feasible. If necessary, we add artificial variables with high costs.

Step 1. Construct and solve the reduced problem $RP_{\mathcal{S}}$.

Step 2. Update $\mathcal{S}$.
This step reduces the degeneracy in RP by removing some zero variables from $\mathcal{S}$ (see Remark 2.2). In practice, updating $\mathcal{S}$ too often destabilizes the RP. See Section 5 for more details.

Step 3. Construct and solve the complementary problem $CP_{\mathcal{S}}$.
The structure of the $CP_{\mathcal{S}}$ constraint matrix is discussed in Section 2.2.

Step 4. If $z^{\text{CP}}_{\mathcal{S}} \geq 0$ then stop; the current solution is an optimal solution of LP. Else augment $\mathcal{S}$ by indices of positive incompatible variables, i.e. $v_j > 0$.
In practice, to accelerate the algorithm we add more than one compatible convex combination. The combination can be aggregated to form an aggregated column as will be explained later.

Step 5. Go to Step 1.

2.2 Compatibility matrix

We introduce below a compatibility matrix that is can be used as a tool to measure to incompatibility (and the compatibility) according to Definition 2.1.

Definition 2.3 *A matrix M is said to be a compatibility matrix if and only if $MA_j = 0$ for every compatible column A_j.*

The matrix M^1, where $M^1 = (A^2_{\mathcal{S}}(A^1_{\mathcal{S}})^{-1}, -I_{m-p})$, $p = |\mathcal{s}|$, and I_{m-p} is an identity matrix of dimension $m-p$, is a compatibility matrix. For any compatible column A_j, $j \in \mathcal{c}_{\mathcal{S}}$, we can easily show that $M^1 A_j = 0$. When $\mathcal{s}$ is nondegenerate, p corresponds to the number of positive variables. This matrix is general in the sense that it is independent of the problem structure. $\|M^1 A_j\|$ is a distance between A_j and its projection (according to M^1) on the vector subspace of compatible columns generated by the columns of $\mathcal{s}$.

There is an infinite number of compatibility matrices. For instance, if P is an invertible matrix and M a compatibility matrix, then PM is also a compatibility matrix because for every compatible column A_j, $MA_j = 0$ is equivalent to $PMA_j = 0$. Compatibility matrix for some SPPs is discussed in Section 4.2.

The choice of the compatibility matrix has a significant impact on the solution process. It is involved not only in identifying the compatible and incompatible columns, but also in computing the coefficients of the CP constraint matrix. When the initial solution (given in Step 0 or found in Step 2 of Algorithm 1) is good, the incompatible columns that are likely to give the greatest improvement in the solution of RP (that is also a solution to LP) are often in a neighborhood "close" to the vector subspace of the compatible columns. To measure the closeness of an incompatible column $j \in \mathcal{z}_{\mathcal{S}}$, we use the notion of the incompatibility degree of A_j. It is denoted $deg_M(j)$ and depends on the compatibility matrix M. If $deg_M(j) = k$ then the column A_j is said to be k-incompatible. The incompatibility degree can be simply defined as the distance of a given incompatible column to the vector subspace of the compatible columns, for instance $deg_M(j) = \|MA_j\|_1$.

This notion of incompatibility degree allows us to implement a multiphase strategy, namely a partial pricing procedure for CP that considers in each phase the incompatible columns with an incompatibility degree (or some measure combining incompatibility degree and reduced cost) not exceeding a certain threshold. These thresholds are increased progressively. In the final phase, we consider all incompatible columns to guarantee optimality.

A more complex partial-pricing strategy for CP would consider at phase k only the incompatible columns A_j with $\frac{\tilde{c_j}}{\|MA_j\|} \leq r_k$ where $\tilde{c_j}$ is the partial reduced cost of incompatible variable j and (r_k) is an increasing sequence of predetermined thresholds. This strategy favors columns with an attractive reduced cost that are "not too far" from the vector subspace of compatible columns. Our numerical results show that this strategy performs well.

3 Improved column generation

When the number of columns is huge or they are not known a priori, we use a CGen decomposition (see Figure 1) that consists in solving an RMP, i.e., a linear program restricted to a subset of variables (columns), yielding a primal solution, possibly feasible, and a dual vector.

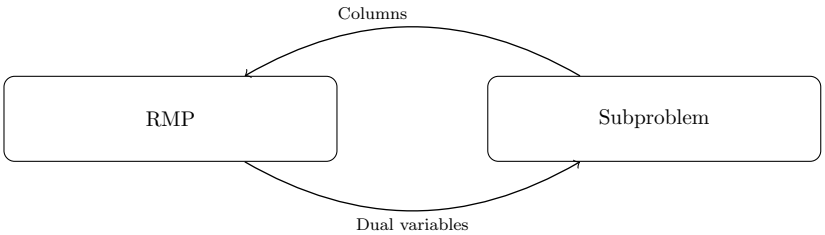

Figure 1: Standard CGen decomposition

The dual vector is used by an oracle (to solve one or many subproblems) to generate columns with negative reduced costs to add to the RMP. The oracle minimizes a reduced-cost function and returns one or many columns with negative reduced costs. We refer to the optimum objective value of the oracle as z_{SP}. The process continues until no variable with a negative reduced cost can be identified.

It is well known that CGen performs poorly in the presence of degeneracy. Starting from the IPS decomposition, we present in Sections 3.1–3.3 three ways of improving the CGen decomposition to overcome the drawbacks of the standard CGen method. These new decompositions permit good dual values without altering the primal feasibility of the solution or increasing the solution time of the RMP. For instance, we may solve RP by a primal method that permits a primal basic feasible solution at each iteration and avoids degeneracy.

3.1 Improving by plugging CGen into IPS

The first approach extends the IPS algorithm to the CGen context, as depicted in Figure 2.

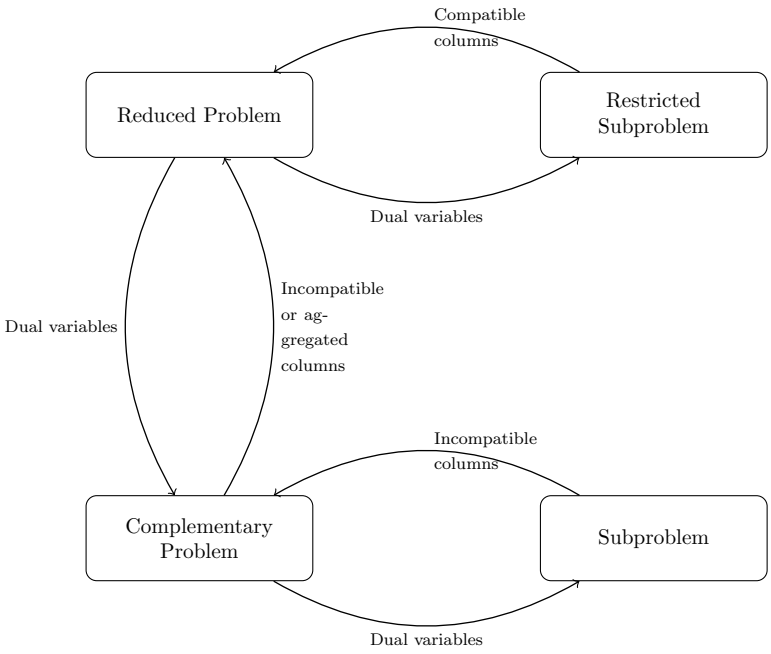

Figure 2: Improved CGen decomposition: plugging CGen into IPS

In Step 1 of Algorithm 1, we solve the RP to optimality using CGen. The subproblem generates only columns that are compatible and have negative reduced costs. In Step 3 of Algorithm 1, we solve the CP to optimality using CGen; this step could be replaced by a call to Algorithm 2.

Algorithm 2 Solving the CP in a CGen context

Step 3.1 Construct and solve the complementary problem $CP_{\mathcal{S}}$.

Step 3.2 If $CP_{\mathcal{S}}$ is optimal, update and return $\mathcal{S}$.
Proposition 3.1 gives the optimality criterion for $CP_{\mathcal{S}}$. Depending on the strategy adopted, $\mathcal{S}$ is updated by adding incompatible columns whose $v_j > 0$ or aggregated columns. See Section 5 for more details.

Step 3.3 Solve the subproblem(s).
For some applications, the subproblems can be solved with a multiphase partial pricing procedure defined as above and discussed later in this paper.

Step 3.4 Augment $\mathcal{I}_{\mathcal{S}}$ and goto Step 3.1.
Observe that no compatible columns should be generated in this step because the RP is solved to optimality (compatible variables have nonnegative reduced costs).

In Step 3.1, the complementary problem $CP_{\mathcal{S}}$ is built and solved. In Step 3.2, if $CP_{\mathcal{S}}$ is optimal and its objective function value is negative then a convex combination (of incompatible columns) is identified.

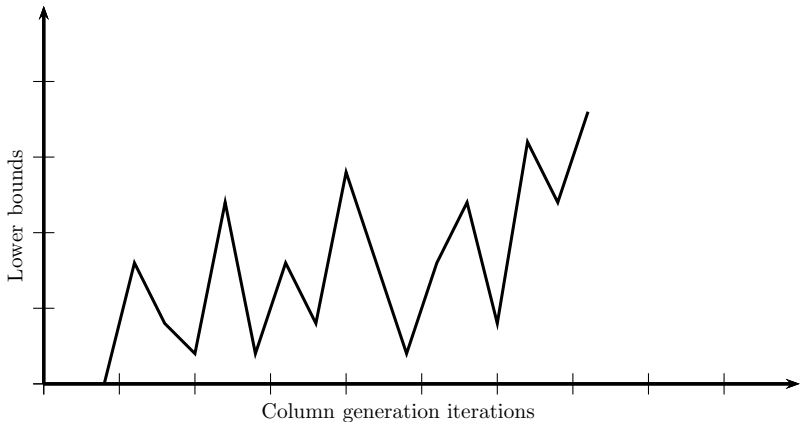

Figure 3: Illustration of the yo-yo effect of the lower bounds

If $CP_{\mathcal{S}}$ is optimal and its objective value is nonnegative, the solution to RP is optimal for the original problem. If $CP_{\mathcal{S}}$ is not optimal then it provides a dual solution for the original problem that is used to solve the subproblem(s) in Step 3.3. The generated incompatible columns are added to $CP_{\mathcal{S}}$ and we return to Step 3.1.

The proposition below discusses the optimality of the CP. Let μ be the dual value related to the convexity constraint (2.13).

Proposition 3.1 *The objective value of the complementary problem is optimal if and only if* $z_{SP} = z_{\mathcal{S}}^{CP} = \mu$.

Proof. The dual solution of CP is optimal if there is no column in the subproblem with a reduced cost smaller than μ. There will always be columns in the subproblem having a reduced cost equal to μ since the columns of CP are present in the subproblem. □

Solving CP to optimality yields the best lower bound on the objective value of LP in a finite number of iterations, under the constraints that the reduced costs of the positive variables of LP are fixed to zero. The following result is valid for all linear programs:

Proposition 3.2 *Let* $\bar{z}$ *be the current objective value of the reduced problem,* $z^\star$ *the value of an optimal solution of the original problem, and* $\bar{y}$ *the value of the objective function of the* $CD_{\mathcal{S}}$. *There exists a constant* K *such that* $\bar{z} + K \cdot \bar{y} \leq z^\star$.

Proof. The best improvement to expect from entering a variable into the basis with a full step not larger than a certain constant, say k, is $k\bar{y}$ and the maximum number of nonzero variables in an optimal basis of LP is m. The current objective value cannot be improved by more than $K \cdot \bar{y}$ where $K = mk$. □

For SPPs, where the maximum value (or step) for a variable entering the basis is 1, we have $K = m$. We can easily build an example where this bound is reached. Consider m columns with reduced cost equal to $\bar{y}$ that form an identity matrix. These columns can all enter the basis and the objective value will be $\bar{z} + m \cdot \bar{y}$. Thus, this lower bound is tight.

Since the minimum reduced cost is maximized in ICG, the bound is stronger and can be used as a heuristic stopping criterion for the CGen. In contrast, in standard CGen the minimum reduced cost fluctuates considerably when the RMP is degenerate. These fluctuations result from the near-random behavior of the dual solutions that jump from one extreme solution to another. The lower bounds on the objective value computed with such reduced costs do not increase monotonically (there is a yo-yo effect), and they are unlikely to reach the maximum value for a given primal solution. Such lower bounds are often useless.

Figure 3 illustrates the yo-yo effect; see also [14], which gives an interesting figure showing the convergence of standard CGen.

The next proposition and corollary explain the yo-yo effect mathematically. In a standard CGen, we pivot on variables with negative reduced costs. Provided we do not enter a variable that could be combined with the previous ones to form a convex combination with a negative reduced cost, the pivots are degenerate. In the best (but unlikely) case, variables forming such a combination are entered one after another. Let us restrict $\mathcal{S}$ to the positive variables in LP and $\mathcal{I}_{\mathcal{S}}$ to k incompatible columns that can be combined to form a minimal compatible convex combination. The set of these columns is denoted $C^1 = \{A_1, \ldots, A_k\}$. The CP, formulated to maximize the minimum reduced cost of these columns, is denoted CD^1.

$$(CD^1) \qquad y^1 = \max_{y,\,\alpha} \; y \tag{3.1}$$

$$\text{s.t.} \quad c_j - \alpha \cdot A_j = 0, \qquad \forall j \in \mathcal{S} \tag{3.2}$$

$$c_j - \alpha \cdot A_j \geq y, \qquad \forall j \in C^1 \tag{3.3}$$

Suppose that $y^1 < 0$. When all the columns of C^1 enter the basis of RP, the objective function value is improved by $\Delta z^1 = y^1(p_1 + \cdots + p_k)$ where $p_j > 0$, $1 \leq j \leq k$, are the values the variables j, $1 \leq j \leq k$, take once pivoted into the basis. Now, instead of entering all the columns of C^1 into the basis, let us enter only the first column A_1 and formulate another complementary problem CD^2 with the remaining columns $C^2 = \{A_2, \ldots, A_k\}$. Since column A_1 cannot on its own improve the objective value, it enters the basis with a degenerate pivot.

$$(CD^2) \qquad y^2 = \max_{y,\,\alpha} \; y \tag{3.4}$$

$$\text{s.t.} \quad c_j - \alpha \cdot A_j = 0, \qquad \forall j \in \mathcal{S} \tag{3.5}$$

$$c_1 - \alpha \cdot A_1 = 0 \tag{3.6}$$

$$c_j - \alpha \cdot A_j \geq y, \qquad \forall j \in C^2 \tag{3.7}$$

Proposition 3.3 *We have $y^2 < y^1$.*

Proof. It is clear that CD^1 is a relaxation of CD^2 and $y^2 \leq y^1$. The optimal solution (α^2, y^2) saturates all the constraints, because, in the primal of CD^2, the columns of C^2 form a minimal convex compatible combination, meaning that the dual values associated to constraints 3.7 that are also the coefficients of the convex combination are positive (> 0). Otherwise, the combination of all the columns of C^1 is not minimal as assumed above because if one coefficient is null there will be a strict subset of columns forming a compatible combination. Therefore, the minimum reduced cost of the columns in C^2 is y^2. When all the columns of C^2 enter the basis of RP, the objective function value is improved by $\Delta z^2 = y^2(p_2 + \cdots + p_k)$. The improvement Δz^2 is equal to Δz^1, and therefore $y^2 < y^1$. □

Corollary 3.4 *Let C^j be the set of columns $\{A_j, \ldots, A_k\}$, $1 \leq j \leq k$, and y^j the largest minimum reduced cost of the columns of the set C^j after pivoting on variables 1 to $j-1$. If the columns of C^j enter the basis by increasing order of index then $y^{j+1} < y^j$, $1 \leq j \leq k-1$.*

Proof. The proof of this corollary follows, by induction, the steps used to prove the proposition. □

We may interpret these two last theoretical results as follows: before we pivot on the last (nondegenerate) variable of the combination, the minimum reduced cost of the nonbasic variables, and consequently the lower bound, strictly decreases. This lower bound generally improves when we pivot on the last nondegenerate variable. This explains the yo-yo effect of degeneracy. It is worse in standard CGen, where we lack the notion of compatible convex combinations and where we pivot on the variables in an almost random way and are unlikely to choose the variables of the best combination.

To find the best lower bound, we must solve CP to optimality via CGen, which could be costly. However, this is not necessary to reach an optimal solution to LP. It suffices to have a convex combination with a negative reduced cost at each iteration to reach optimality for the original problem. We can stop the CP solution process heuristically, defining a convex combination to be good when $z_{\mathcal{S}}^{CP} - z_{SP} \leq \epsilon$, where ϵ is a tolerance.

This decomposition is closer to the one presented in [4]. A direct implementation of this decomposition do not work efficiently according to our preliminary tests because, as we observed, the complementary problem is too degenerate and solving it by column generation would suffer a lot. This decomposition needs using some sophisticated techniques (stabilization of CP for instance) to fully benefit from it.

3.2 Improving by plugging IPS into CGen

Another way to improve the CGen approach is to solve the RMP by IPS (Algorithm 1) to deal with degeneracy by improving the quality of the dual solutions. IPS decomposition introduces a supplementary level corresponding to CP; this decomposition is illustrated in Figure 4. The RMP is decomposed into an RP and a CP (restricted to the generated columns) to be solved with IPS. At each minor iteration, the RP is solved and its dual solution is passed to CP to find a convex combination of the incompatible columns to improve the objective value of the RP. If the objective value of CP is nonnegative then the RP solution is optimal for RMP, and the dual solution given by CP is used to generate new columns.

An interesting result of this improved CGen concerns the dual-solution quality given by CP in the CGen context. This result is proved from the $CD_{\mathcal{S}}$ formulation given by (2.15)–(2.17). Let (α, y) be a solution of $CD_{\mathcal{S}}$.

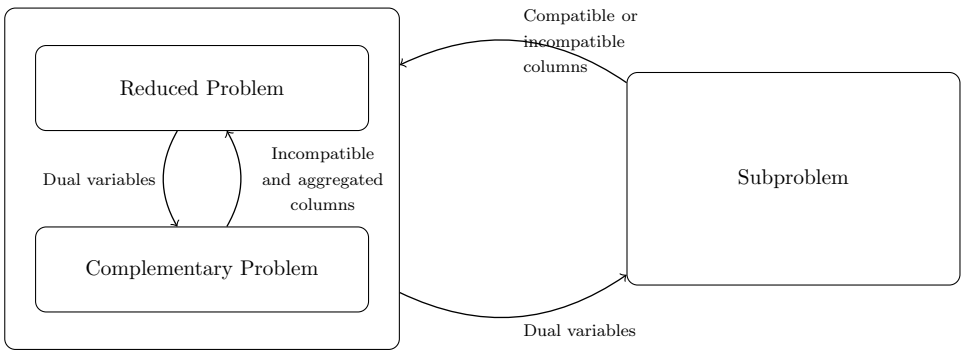

Figure 4: Improved CGen decomposition: plugging IPS into CGen

Theorem 3.5 *If $z_{\mathcal{S}}^{CD} > 0$ and $|\mathcal{S}| < m$ then α is a nonbasic dual solution for the original problem.*

Proof. We start by showing that α is dual feasible for the original problem, and then we show that it is nonbasic. The solution α is feasible if the reduced costs of all the columns are nonnegative. The set of all columns J is partitioned into three subsets: $\mathcal{S}$, $\mathcal{C}_{\mathcal{S}} \setminus \mathcal{S}$, and $\mathcal{I}_{\mathcal{S}}$. The columns of $\mathcal{I}_{\mathcal{S}}$ have positive reduced costs. The reduced costs of the columns of $\mathcal{S}$ are zero. As shown in [7], any solution of 2.16 makes the reduced cost of $\mathcal{C}_{\mathcal{S}} \setminus \mathcal{S}$ nonnegative. It follows that α is dual feasible for the original problem.

If α is a basic dual solution then there exists at least one primal basis B of dimension m that can be associated with it. We show that such a solution does not exist. Recall that the reduced cost of any basic variable is zero. The variables $\mathcal{S}$ are linearly independent and their reduced cost is zero, so they can to be in the basis B, forming its first part $B_{\mathcal{S}}$. All the incompatible columns have positive reduced costs so they cannot be in the basis. Compatible columns cannot be used to complete $B_{\mathcal{S}}$ since they are linearly dependent on $\mathcal{S}$. Since $B_{\mathcal{S}}$ cannot be completed to form a basis B, the solution α is nonbasic. □

In practice, the dual solutions of the original problem produced by $CD_{\mathcal{S}}$ are interior. This leads to the stability of the dual solutions, as discussed in Section 5. Unlike the extreme dual solutions in standard CGen,

central dual solutions do not zigzag from one extreme point to another of the dual polyhedron. Central dual solutions drastically reduce the number of iterations. This is the reason for using primal-dual methods such as the interior point (barrier) method. The advantage of ICG is that we do not face the numerical difficulties encountered in the barrier method. A long tail is generally observed in the latter when the solution becomes close to optimal, and the solution is very fractional and requires posteriori treatment before branching.

3.3 Improving with dynamic variable aggregation

In this subsection, we discuss the benefits of dynamic variable aggregation in CGen. Pivoting on a single incompatible column with a negative reduced cost, even if it is in a solution to $CP_{\mathcal{S}}$, leads to a degenerate simplex pivot. A solution to $CP_{\mathcal{S}}$ can be aggregated to form an aggregated column (variable) as mentioned previously. This column is compatible by construction. Therefore, adding it to RP does not require any change in the RP basis size, leading to greater numerical stability.

Consider the set $\Omega_{\mathcal{S}}$ defined below for $\mathcal{S}$ derived from an optimal solution to RP: $\Omega_{\mathcal{S}} = \{\omega \in \Omega \mid \omega$ is compatible with $\mathcal{S}$ and $\bar{c}_\omega < 0\}$ where $\Omega = \left\{\omega = \sum_{j\in\mathcal{I}_{\mathcal{S}}} v_j A_j \mid \sum_{j\in\mathcal{I}_{\mathcal{S}}} v_j = 1, v_j \geq 0\right\}$, $\bar{c}_\omega = \sum_{j\in\mathcal{I}_{\mathcal{S}}} v_j \tilde{c}_j$, and $\tilde{c}_j = \left(c_j - c_{\mathcal{S}}^\top \left(A_{\mathcal{S}}^1\right)^{-1} A_j^1\right)$ is the *partial* reduced cost of incompatible variable j computed according to the constraints of RP. Let $\mathcal{I}_\omega = \{j \in J | v_j > 0\}$. An element $\omega \in \Omega_{\mathcal{S}}$, if exists, is a negative-reduced-cost, compatible, convex combination of (incompatible) columns that forms an aggregated column. It is easy to see that $\Omega_{\mathcal{S}} \neq \emptyset$ if and only if $z_{\mathcal{S}}^{\text{CP}} < 0$. Pivoting on the aggregated variable decreases the objective value, as the next proposition shows.

Proposition 3.6 *If $\mathcal{S}$ is nondegenerate, then pivoting on any aggregated variable with a negative reduced cost decreases the objective value of the reduced problem.*

Proof. The proof is quite intuitive. If we enter the convex combination as an aggregate compatible variable (with a negative reduced cost), since the basis of the RP is nondegenerate, the objective value will decrease. Pivoting on ω or on the variables indexed by $\mathcal{I}_\omega$ has the same impact on the objective function value. This is shown in [13]. □

By its structure, CP is easier to solve than RP, especially when the multiphase strategy is used. It is better to keep columns in CP rather than increasing the basis size of RP. From this observation we derive the following strategy: 1) Use the aggregated variables to improve the dual solution. This allows us to handle a large number of incompatible columns (in CP that seems to be the right place) and thus stabilizes the dual solutions without increasing the size of RP. 2) Generate "fresh" columns from the subproblem to augment $\mathcal{S}$ (slightly), i.e., increase the basis size of RP and hence improve the objective value more substantially. Another observation about the aggregated columns is that they do not stay long in the basis of RP. Generally, more attractive original columns generated in the subsequent iterations replace them. The number of positive aggregated variables decreases at the end of the CGen process. Interestingly, even if we have aggregated variables in the final basis, the final solution corresponds to an extreme point of the original problem, as the next proposition shows.

Proposition 3.7 *Variables involved in an optimal basic solutions with aggregated variables have their reduced cost equal to 0.*

Proof. Each basic variable, aggregated or not, has a null reduced cost in an optimal solution. Variables forming an aggregated variable have nonnegative reduced cost because we have an optimal solution. As the aggregated variable has a null reduced cost and the coefficients of the combination are positive ($v_j > 0$), the individual variables forming the aggregated variables have a null reduced cost. □

From Proposition 3.7, we can easily derive a basic primal solution. Having an extreme point at the end of the solution process is useful for branching or cutting. We do not need a crossover procedure, often used

in interior point methods, to obtain an extreme point. In Section 5, we illustrate numerically the percentage of positive aggregated variables in the final basis of the RP.

4 Improved dynamic constraint aggregation

As mentioned earlier, we specialize ICG to take advantage of the beautiful structure of SPPs. We show that this specialized version is a generalization of DCA and MPDCA, and we refer to it as improved dynamic constraint aggregation (IDCA).

An SPP is a special case of LP where x is binary, A is a $0-1$ matrix, and b is $e = (1, \ldots, 1)^\top$. When we relax the binary constraint on x, we obtain a linear relaxation of the SPP. It is this linear relaxation that we address in this section.

Suppose that we have a set T of m elements, say a set of tasks (such as flights or bus trips). The nonzero elements of each column of A form a feasible subset of T with a given cost. Here, feasible means that we can find a column that covers the subset. The (set partitioning) constraints indicate that each element (task) of T must belong to exactly one selected subset. We want to minimize the cost of the selected subsets.

We use the word *cluster* to refer to a feasible subset of tasks. Denote by L an index set of clusters and by T_ℓ the l^{th} cluster. $Q = \{T_\ell : \ell \in L\}$ is a partition of T if $T_{\ell_1} \cap T_{\ell_2} = \emptyset$ for all $\ell_1, \ell_2 \in L$, $\ell_1 \neq \ell_2$, and $\bigcup_{\ell \in L} T_\ell = T$. Because the tasks in each cluster are aggregated in DCA and MPDCA, we call it *an aggregating partition*. We often omit the word *aggregating* for the sake of simplicity. In [6], we used a definition of compatibility with respect to an aggregating partition similar to the following.

Definition 4.1 *A column $A_j, j \in J$ is said to be* compatible *with cluster $\ell \in L$ if it covers all elements in T_ℓ or none of them. If a column A_j is compatible with all clusters in L, then A_j and its associated variable x_j are said to be* compatible *with partition Q. The columns and variables that are not compatible with Q are said to be* incompatible.

Note that if a column is compatible with a partition Q, then the subset of tasks covered by this column is the union of a subset of clusters of Q. Definition 4.1 is a special case of Definition 2.1.

4.1 Links between IDCA, DCA, and MPDCA

In DCA and MPDCA, we build the RP by aggregating the set partitioning constraints corresponding to the tasks belonging to the same cluster. We refer to the RP as the aggregated restricted master problem (ARMP) in this context. By solving this ARMP, we obtain aggregated dual values for the aggregated set partitioning constraints, i.e., an aggregated dual value for each cluster. To price out incompatible columns, especially in the CGen context, we need an individual dual value for each set partitioning constraint (task). To efficiently disaggregate these aggregated dual values, we try to satisfy the dual constraints related to incompatible columns by solving the linear system:

$$\sum_{i \in T_\ell} \alpha_i \quad = \hat{\alpha}_\ell, \qquad \forall \ell \in L \tag{4.1}$$

$$\sum_{i \in T} a_{ij} \alpha_i \leq c_j, \qquad \forall j \in \mathcal{I}_Q \tag{4.2}$$

where $\mathcal{I}_Q$ is the index set of columns incompatible with partition Q, α_i is the dual variable associated with the set partitioning constraint i, and $\hat{\alpha}_\ell$ is the aggregated dual value of cluster l computed at the aggregated problem level (ARMP).

In [5, 6], we assumed that the tasks (set partitioning constraints) are ordered, which is the case in many vehicle and crew scheduling problems. For instance, the tasks may be ordered by their start times. In this section, we consider only the ordered case for comparison with DCA and MPDCA. In the following section, we discuss partially ordered and unordered tasks.

By means of a change of variables, we introduced in [5, 6] a new set of variables π_ℓ^h, $h \in \{1, \ldots, |T_\ell|\}$ and $\ell \in L$, defined by

$$\pi_\ell^h = \sum_{i=1}^{h} \alpha_\ell^i, \qquad \forall h \in \{1, \ldots, |T_\ell|\},\ \ell \in L \tag{4.3}$$

where α_ℓ^i is the dual variable of the set partitioning constraint associated with the i^{th} task of cluster ℓ. The double index is used to specify the order of the task in cluster ℓ. With this change of variables, equations (4.1) disappear because the corresponding π variables become constants. In DCA and MPDCA, after the variable change, only the constraints corresponding to difference inequalities, i.e., in the form $\tilde{c_j} - (\pi_{\ell_1}^{h_1} - \pi_{\ell_2}^{h_2}) \geq 0$, are kept in the system. Here $\tilde{c_j}$ is the partial reduced cost of variable j, i.e., the reduced cost computed according to the constraints of the ARMP.

The resulting linear system could be represented by a network G where the h^{th} task of cluster ℓ is represented by a node n_ℓ^h. Each difference constraint of the linear system, and hence each corresponding incompatible variable (column), is represented by an arc from node $n_{\ell_2}^{h_2}$ to node $n_{\ell_1}^{h_1}$. The cost of the arc is the partial reduced cost $\tilde{c_j}$. The "dual" of this linear system is a flow in the network G. A detailed discussion on this network representation can be found in [5]. Solving this linear system is equivalent to finding a shortest path in G. We denote these constraints and the corresponding *special* incompatible columns (arcs) by $\mathcal{I}_{\mathbb{S}}$. We may also add some artificial arcs to connect the graph if necessary. We solve this shortest path problem with Bellman's algorithm to find a dual solution of the original problem to price-out incompatible columns from the subproblem. If a negative cycle is detected, we conclude that the linear system is infeasible, meaning that one or more dual constraints are infeasible. Actually, a negative cycle means the network flow problem (shortest path problem) is unbounded due to the presence of a (negative) extreme ray, and consequently its dual, i.e. the linear system, is infeasible.

To avoid these negative cycles, some of the arcs are removed from G so that we can find a feasible solution to the remaining subsystem. Proposition 4.3 is a new result showing that negative cycles help to improve the objective value of ARMP and therefore we do not have to avoid them. In contrast to MDCA and DCA, where it is difficult to manage negative cycles, IDCA treats them easily, and we present a criterion in Proposition 4.6 for ranking them.

In IDCA, we combine IPS, DCA, and MPDCA. The idea is to replace the disaggregation heuristic procedure of dual variables of [5, 6] by an exact appoach. The RP is exactly the same ARMP as in DCA and MPDCA. Each cluster of tasks can be seen as an artificial column of a cost equal to the dual value of the corresponding aggregated constraint. Constraints (2.16) are replaced by Constraints (4.1), giving the following formulation of the CP:

$$(CD_Q) \qquad z_Q^{CD} = \max_{y,\ \alpha} \quad y \tag{4.4}$$

$$\text{s.t.} \quad \sum_{i \in T_\ell} \alpha_i = \hat{\alpha}_\ell, \qquad \forall \ell \in L \tag{4.5}$$

$$c_j - \alpha \cdot A_j \geq y, \qquad \forall j \in \mathcal{I}_Q \tag{4.6}$$

When the change to π variables is applied, Constraints (4.5) disappear because the corresponding π variables become constants. To present some theoretical results about the IDCA links to DCA and MPDCA, we restrict $\mathcal{I}_Q$ to the special incompatible columns $\mathcal{I}_{\mathbb{S}}$ considered in the shortest path problem of DCA/MPDCA. Note that in IDCA we consider all the incompatible columns since a linear problem rather than a shortest path problem is used to disaggregate the dual variables. The inequalities of the restricted $\mathcal{I}_{\mathbb{S}}$ are transformed into difference inequalities to formulate a restricted CD_Q as follows.

$$(RCD_Q) \qquad y_{opt} = \max_{y,\ \pi} \quad y \tag{4.7}$$

$$\text{s.t.} \quad \tilde{c_j} - (\pi_{\ell_1}^{h_1} - \pi_{\ell_2}^{h_2}) \geq y, \qquad \forall j \in \mathcal{I}_{\mathbb{S}} \tag{4.8}$$

where $\tilde{c_j}$ is the partial reduced cost of variable j.

Remark 4.2 *The cost of an arc in G corresponds to the partial reduced cost of its corresponding column in IDCA.*

It is easy to see that if we require y to be 0 in RCD_Q we obtain the graph G where the cost of the arc will be $\tilde{c_j}$. Similarly, if we consider the dual of RCD_Q, i.e., RCP_Q, its cost vector component will be $\tilde{c_j}$. Theoretical results on the links between IDCA and DCA/MPDCA are presented below.

Proposition 4.3 *Each negative cycle in G corresponds to a convex combination (of incompatible columns) having a negative reduced cost. When these columns are added to a nondegenerate ARMP, the objective function value decreases.*

Proof. Let F be a cycle with a negative cost composed of q nodes (representing tasks) indexed by $\mathcal{I}_{\mathbb{S}} = \{i_1, i_2, \ldots, i_q\}$. The j^{th} arc of the cycle is represented by the pair (i_j, i_{j+1}) for $j \in \{1, ..., q-1\}$, and the q^{th} arc is represented by the pair (i_q, i_1). Let us show that the optimal value of RCD_Q restricted to only the cycle F corresponding to the dual constraints is negative. Constraints (4.8) of RCP_Q become:

$$\begin{aligned} y_{opt} &\leq \tilde{c_{i_1}} - (\pi_{i_2} - \pi_{i_1}) \\ y_{opt} &\leq \tilde{c_{i_2}} - (\pi_{i_3} - \pi_{i_2}) \\ &\vdots \\ y_{opt} &\leq \tilde{c_{i_q}} - (\pi_{i_1} - \pi_{i_q}). \end{aligned}$$

The sum of these constraints gives $y_{opt} \leq \frac{1}{q} \sum_{j=1}^{q} \tilde{c_{i_j}}$. Since the cycle has a negative cost, $y_{opt} < 0$. From Theorem 3.6, when we add the incompatible columns, corresponding to the arcs of cycle F, to ARMP assumed to be nondegenerate, the objective function value decreases. □

Proposition 4.4 *Each elementary cycle F in G, in the sense that we cannot extract a subcycle of F, is a minimal convex combination.*

Proof. Let F be an elementary cycle. In 4.8, we restrict $\mathcal{I}_S$ to the incompatible columns corresponding to only the arcs of F. We can easily show that if we remove a constraint from RCD_Q, y_{opt} becomes unbounded. This means that no convex combination of a strict subset of incompatible columns can be found. In other words, the cycle F corresponds to a minimal convex combination. □

The following corollary is interesting because it improves the DCA and MPDCA algorithms. We formerly believed that for DCA and MPDCA finding a negative cycle is problematic, but this is not true. The proof of this corollary is immediate.

Corollary 4.5 *When we run a shortest path algorithm on G, we either:*

1. *detect a negative cycle (convex combination with a negative reduced cost) that decreases the objective value of a nondegenerate ARMP, or*
2. *obtain a dual solution that is feasible for the incompatible columns represented in G, indicating that the current primal solution is optimal for the compatible and incompatible columns represented in G.*

This corollary generalizes a result of [6] that gives conditions under which the addition of **two** special incompatible variables to the basis ensures a decrease in the objective value.

Let us now suppose that the optimal value of the problem RCP_Q is negative ($y_{opt} < 0$) and we have n possible cycles F_1, ..., F_n of negative cost. Let $\bar{C}_i$ be the cycle (reduced) cost and $|F_i|$ the number of arcs in cycle F_i. Which would be the promising cycle output by the RCP_Q?

Proposition 4.6 *The cycle corresponding to* $\min_i \frac{\bar{C}_i}{|F_i|}$ *is the cycle output by the complementary problem RCP_Q.*

Proof. We show that $y_{opt} = \min_i \frac{\bar{C}_i}{|F_i|}$. It is easy to see that $y_{opt} \leq \min_i \frac{\bar{C}_i}{|F_i|}$ by summing the Constraints (4.8) corresponding to the cycle i as in the proof of Proposition 4.3. Since $y_{opt} < 0$, the resolution of problem RCP_Q gives a cycle with negative cost corresponding to $\min_i \frac{\bar{C}_i}{|F_i|}$. Consequently, $y_{opt} = \min_i \frac{\bar{C}_i}{|F_i|}$. □

By ordering the cycles using this ratio, we give a better rating to small cycles and disaggregate the partition less when we enter these variables into ARMP. This is minimum-mean cycle. The reader may refer to [9] (a recent paper on this topic) for more details.

4.2 Compatibility matrix for some structured SPPs

Recall the purpose of the compatibility matrix: it tells us whether or not a column is compatible and measures the degree of incompatibility of this column. A good compatibility matrix would use the structure of the problem to generate good columns with low degrees of incompatibility and thus reduce the number of nonzero elements in the CP constraint matrix.

For vehicle and crew scheduling problems where the crew members usually follow the vehicle routes, the matrix M^2 below seems to be an excellent compatibility matrix. For a problem where this property is not preserved M^2 may be less appropriate. M^2 measures the deviation from the sequencing of tasks in the clusters (e.g., vehicle routes). The tasks in these problems can be totally ordered by starting time for instance.

$$M^2 = \begin{pmatrix} M_1^2 & & & 0 \\ & M_2^2 & & \\ & & \ddots & \\ 0 & & & M_{|L|}^2 \end{pmatrix}$$

where

$$\left(M_\ell^2\right)_{\substack{1 \leq i \leq T_\ell - 1 \\ 1 \leq j \leq T_\ell}} = \begin{pmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -1 & 0 \\ 0 & \cdots & 0 & 1 & -1 \end{pmatrix}$$

The compatibility matrix M^2 corresponds to the change of variables mentioned above in Constraints (4.5). When an incompatible column A_j does not respect a cluster sequence, a nonzero coefficient (-1 or $+1$) appears in $M^2 A_j$. The coefficient -1 indicates that A_j covers a task but not its predecessor, and $+1$ indicates that A_j covers a task but not its successor. The number of nonzero elements of $M^2 A_j$ corresponds to the number of clusters to be added (by breaking up current clusters) to make A_j compatible. The value $deg_{M^2}(j)$ counting the number of nonzeros in $M^2 A_j$ measures the deviation of A_j from the ordered clustering and is a good measure for the multiphase partial pricing strategy.

The following mathematical affirmations (4.7–4.9) demonstrate that the definition of the degree of incompatibility corresponds to the concept used in MPDCA. In MPDCA, we essentially added to DCA a partial pricing strategy controlled by phases. In this strategy, we consider at phase k incompatible columns with at most k incompatibilities (as defined below) with the current solution. This strategy reduces the computational time per iteration by keeping ARMP smaller and reducing the number of iterations. In MPDCA, we introduced a k-incompatibility concept as follows:

Definition 4.7 *A column is said to be k-incompatible if we have to add k additional clusters to the partition to make this column compatible (with the new partition).*

The next proposition establishes a more formal link between this concept and the matrix M^2.

Proposition 4.8 *A column A_j, $j \in \mathcal{I}_Q$, is k-incompatible if and only if $M^2 A_j$ has k nonzero (1 or -1) elements.*

Proof. We show this property by induction. It is true for 0-incompatible columns (i.e., compatible columns): if A_j is compatible then by definition $M^2 A_j = 0$. Thus, $M^2 A_j$ has 0 nonzero elements. We assume that the property is true for k and fewer incompatibilities. Suppose that A_j has $k+1$ incompatibilities. A_j could then be rewritten as $A_j^1 + A_j^2$ where A_j^1 and A_j^2 are incompatible and orthogonal, and A_j^1 covers the first block of consecutive tasks of A_j. In fact, A_j^1 has r incompatibilities ($r = 1$ or 2), so A_j^2 has $k+1-r$ incompatibilities. By the assumption, $M^2 A_j^1$ has r nonzero elements and $M^2 A_j^2$ has $k+1-r$ incompatibilities. We can easily prove that $M^2 A_j^1$ and $M^2 A_j^2$ are disjoint. Thus, $M^2 A_j$ has $k+1$ nonzero elements. □

Corollary 4.9 *In phase k, we have at most $(k+1) \cdot |\mathcal{I}_Q|$ nonzero coefficients in the constraint matrix of the complementary problem.*

Proof. This can be deduced from Proposition 4.8 and the fact that each incompatible column contributes 1 to the convexity constraint. □

This is a good compatibility matrix because it ensures sparsity in the columns of the CP that are in a close neighborhood of the vector subspace of compatible columns. These columns also have the potential to improve significantly the objective value of RP. In phase k, we consider only the l-incompatible columns for l less than or equal to k. Of course, the lower the phase number, the shorter the distance to the vector subspace of compatible columns and the less dense the columns of CP. The columns with good potential are considered first, and the computational cost of pivoting in these columns, for both RP and CP, is reduced. The efficiency of this matrix is discussed in Section 5.

4.3 IDCA algorithm

The algorithm proposed in this section is an adaptation of the decomposition framework described in Sections 3.2 and 3.3 to the special case of the SPP. We require sophisticated techniques to fully profit from the decomposition presented in Section 3.1. In IDCA, we combine the decompositions of Sections 3.2 and 3.3 simply because this is easier to implement.

We adapt Algorithm 1 as follows. In Step 0, it is not necessary to provide an initial solution; an initial partition of the set partitioning constraints (tasks) Q suffices. For example, in vehicle and crew scheduling problems, the set Q refers to trips. Each vehicle performs a sequence of trips, and drivers generally change vehicles just a few times during a working day. The changes made to Step 1 of Algorithm 1 are reported in Algorithms 3 and 4. The partition Q is updated each time $\mathcal{S}$ is updated, as shown in Algorithm 4. The ARMP is built as indicated in Algorithm 3 by removing some redundant (only identical) constraints and adding variables that are compatible with the updated partition. The major advantages of this reduction technique are its simplicity and speed. Its only drawback is that the reduced basis may contain degenerate basic variables. In this case, we cannot theoretically guarantee that when we add a negative convex combination to RP the objective function value will decrease. However, in practice the number of degenerate basic variables is small, so the negative impact of these variables is minimized. The ARMP is then solved to obtain an improved primal solution and also to provide CP with an aggregate dual solution.

Algorithm 3 Specialization of Step 1 of Algorithm 1

Step 1.1	Update partition Q using Algorithm 4.
Step 1.2	For each cluster keep one constraint in ARMP and add all the columns that are compatible with Q.
Step 1.3	Solve ARMP.

Algorithm 4 Partition Update

Step 2.1 For each task $t \in T$, mark all the columns j in $\mathcal{S}$ covering t.
Step 2.2 Put all the tasks covered by the same columns in the same cluster.

We do not update $\mathcal{S}$ in Step 2 of Algorithm 1 for the test problems presented in this paper. No reaggregation is needed since the size of ARMP does not increase much: 1) few partition disaggregations are needed because we favor compatible variables and perform variable aggregation. In a minor iteration (between ARMP and CP), if the objective function value of CP is found to be negative, its solution is aggregated into a compatible aggregated column and this column is added to ARMP. 2) Disaggregation is made with variables with low incompatibility degrees. 3) Some degeneracy is allowed in ARMP to avoid frequent changes to its basis size.

The CP is defined in Step 3 of Algorithm 2 using the compatibility matrix M^2. The CP constraint matrix is sparse, and this sparsity allows the CPLEX pre-solver to considerably reduce the size of CP. A numerical justification of this choice is given in Section 5. The dual simplex is more appropriate to solve CP.

We use the multiphase concept as in MPDCA, starting of course from phase 1. At each iteration of the CGen, the subproblem is solved to generate columns with an incompatibility degree that does not exceed $Phase$, as indicated in Algorithm 5. The parameter $Phase$ is progressively incremented and reaches $PhaseMax$ in the final phase. In the final phase we consider all possible incompatible columns in order to find an optimal solution.

Algorithm 5 Multiphase approach for the subproblem

Solve the subproblem(s) in a given $Phase$.
If no column with negative reduced cost is generated and $Phase < PhaseMax$ then set $Phase = Phase + 1$ and loop.

Note that the subproblem is a shortest path problem with resource constraints [11]. The implementation of the multiphase strategy is accomplished by simply adding a resource constraint to the subproblem that counts the number of incompatibilities. This is done by an extension function that increases this number each time we break a task sequence of a cluster. This resource is upper bounded by the phase number: in phase k, only columns with a maximum incompatibility degree of k are generated. This resource makes the subproblem easier to solve when k takes small values. In fact, small values of k decrease the number of labels kept in memory when we solve the subproblem with dynamic programming. Generally, an optimal or near-optimal solution is found in phase 2.

We do not decrease the phase number during the solution process, so compatible columns may be generated along with the incompatible ones. If the negative reduced costs of the incompatible columns are more interesting, we treat them first by adding some of them to $\mathcal{S}$. We thus disaggregate the partition. If the negative reduced costs of the compatible columns are more interesting, we treat them first by adding them to ARMP; no disaggregation is done in this case. A parameter is used to decide whether or not to disaggregate the partition. This is the only parameter required for the implementation of IDCA in a CGen framework.

5 Computational experiments

In this section, we compare the computational performance of standard column generation (SCG) and IDCA on the linear relaxation of the simultaneous vehicle and crew scheduling problem (VCSP). Given a set of timetabled bus trips and a set of possible driver relief points, VCSP finds an optimal schedule for both the drivers (crews) and buses (vehicles). The simultaneous vehicle and crew scheduling problem is formulated as an SPP with side constraints. The set partitioning constraints ensure that each trip segment is covered by exactly one driver. The side constraints allow us to find a posteriori, in polynomial time, an optimal bus schedule. In a valid VCSP solution, every trip is covered by a bus and the crew schedules satisfy the

collective agreement. The driver schedules (columns) are generated by a subproblem (SP) that implements these collective agreement rules. For a detailed description of this problem, see [10].

The instances of VCSP used here are those reported in [6]. The tests were performed on a DELL 64-bit Oracle Linux machine (Intel i7, four CPU cores, 3.4 GHz). We use version 4.5 of the column generation software GENCOL (commercialized by Kronos Inc.), which uses CPLEX version 12.4 to solve RMP, ARMP, and CP.

5.1 Numerical results

We first consider the number of CGen iterations required to solve the largest instances of VCSP, as reported in Table 5.1. The first column gives the algorithm used, the next two give the CPLEX version and solver, and the remaining columns give the iterations required to solve VCSP instances with 1200, 1600, and 2000 set partitioning constraints.

Table 5.1: Number of CGen iterations required for the largest VCSP instances

	CPLEX version	CPLEX solver	VCSP instances		
			1200	1600	2000
GENCOL 4.3	7.5	primal	480	618	798
GENCOL 4.5	12.4	primal	545	554	710
GENCOL 4.5	12.4	dual	331	372	464
MPDCA & GENCOL 4.3	7.5	primal	226	433	242
IDCA & GENCOL 4.5	12.4	primal	83	98	135
IDCA & GENCOL 4.5	12.4	dual	90	92	117

We choose the primal solver in CPLEX version 7.5 because it is better than the dual solver in that version of CPLEX. However, in version 12.4 the dual solver is 1.6 times faster than the primal. The results indicate that the CPLEX version has a greater influence than the GENCOL version on the number of CGen iterations. The IDCA algorithm reduces the number of iterations by a factor of almost 4 on average compared to GENCOL with CPLEX 12.4 when the dual solver is used and by a factor of 6 when the primal solver is used. The time per iteration is reduced too by factor exceeding 2 on average as can be seen in the tables that follow. Solving ARMP in IDCA with the CPLEX primal or dual solvers does not affect the behavior of the CGen process because ARMP is nondegenerate or only slightly degenerate, and the dual solutions obtained by the IDCA decomposition are good. The IDCA algorithm not only reduces the number of iterations compared to MPDCA but is also more stable. We cannot expect great improvements in MPDCA if the solver is changed to the dual because of the aggregation of the RMP. For these reasons and because of the unavailability of the GENCOL 4.3 libraries, we focus on comparing IDCA to standard column generation (referred to as SCG).

The IDCA requires an initial aggregated partition of the tasks that we built in the same way as in [6]. The initial partition contains as many clusters as trips; each cluster is a sequence of tasks (trip segments). It was shown in [6] that the quality of the initial partition has a significant impact on the computational time for the multiphase strategy. A good initial partition allows us to reach optimality with a relatively small number of constraints in ARMP, which reduces the proportion of degenerate variables in the basis. This is confirmed by IDCA in this paper.

The results for small and medium VCSP instances (up to 800 tasks) are reported in Table 5.2, and the results for large instances (up to 2000 tasks) are reported in Table 5.3. For each algorithm (CG and IDCA), we provide the computational time (Time), the number of CGen iterations (CgIter), the number of minor iterations (MinorItr), the computational time for RMP or ARMP (RmpT), the computational time for CP (CpT) and SP (SpT), the average number of columns of RMP/ARMP (RmpC) and CP (CpC), and finally the size of the partition (the number of set partitioning constraints in RMP or ARMP) when optimality is reached. All times are given in seconds.

The results show that IDCA reduces the computational time on average by a factor of 9 compared to SCG. This is mostly a result of the reduction that IDCA achieves in the solution time of the master problem;

Table 5.2: Numerical results for small and medium instances

Algorithm	SCG			IDCA		
Instance	400	600	800	400	600	800
Time	15.07	79.70	150.10	1.43	8.93	12.63
CgItr	156	272	233	35	57	66
MinorItr	–	–	–	26	94	70
RmpT	7.37	36.43	74.63	0.17	1.93	1.77
CpT	–	–	–	0.30	2.80	3.33
SpT	7.63	42.90	74.87	0.77	3.57	6.57
RmpC	1444	2287	3025	206	634	811
CpC	–	–	–	1042	2411	3599
PartSize	400	600	800	64	124	156

Table 5.3: Numerical results for large instances

Algorithm	SCG			IDCA		
Instance	1200	1600	2000	1200	1600	2000
Time	644.23	1609.13	4281.67	77.47	228.87	516.43
CgItr	331	372	467	90	92	117
MinorItr	–	–	–	170	280	286
RmpT	343.53	956.50	2538.33	13.23	41.73	102.57
CpT	–	–	–	31.83	115.10	221.03
SpT	299.03	649.60	1737.97	28.47	63.17	168.90
RmpC	4749	6405	8150	1912	3529	5714
CpC	–	–	–	8236	14223	17809
PartSize	1200	1600	2000	264	336	468

IDCA reduces this time by a factor of 35 (on average) for the small and medium instances and a factor of 25 for the large instances. If we consider that in IDCA solving both ARMP and CP is equivalent to solving RMP in SCG, the reduction factor is approximately 8. These results reflect the quality of both the dual solutions provided by CP and the compatibility matrix, as discussed below.

5.2 Numerical efficiency of compatibility matrix

The compatibility matrix is used to define a partial pricing procedure in SP and to build the CP constraint matrix. Therefore, choosing the right matrix is crucial since it affects the whole algorithm. A good compatibility matrix allows us to easily implement the partial pricing procedure in SP and to generate high-quality columns in the first phase. Such columns, when added to CP, lead to a sparse constraint matrix and a solution (a convex compatible combination of incompatible variables) that does not disaggregate ARMP too much. The compatibility matrix M^2 satisfies these criteria, as shown below.

The implementation (in the subproblem) of the multiphase strategy induced by the compatibility matrix M^2 is easy and efficient, as described in Section 4.3.

The densities of the CP constraint matrix in IDCA and the RMP constraint matrix in SCG are reported in Figure 5 for a medium VCSP instance (800 tasks). Each time CP or RMP is solved, its density is reported. The density of RMP grows rapidly during the first iterations because the artificial columns (added to RMP to ensure its feasibility) are replaced by dense columns. When the dual variables of RMP are sufficiently stable, its density grows more slowly. The density of CP is stable around its average (0.3%), which makes it easier to solve than RMP and ARMP. CP is 10 times less dense than RMP. The average solution time per iteration is 0.34 for RMP, 0.02 for ARMP, and 0.01 for CP. CP is generally solved more than once at each iteration to find orthogonal, in the sense of row-disjoint, combinations of columns to add to ARMP, i.e., "orthogonal" descent directions. Even so, the total time spent solving CP is on average 15 times less than the time spent solving RMP in SCG and only twice as long as the time spent solving ARMP.

The partition size in the last iteration of IDCA is a measure of the efficiency of the chosen compatibility matrix: the smaller the partition size at optimality, the better the compatibility matrix. For IDCA, the

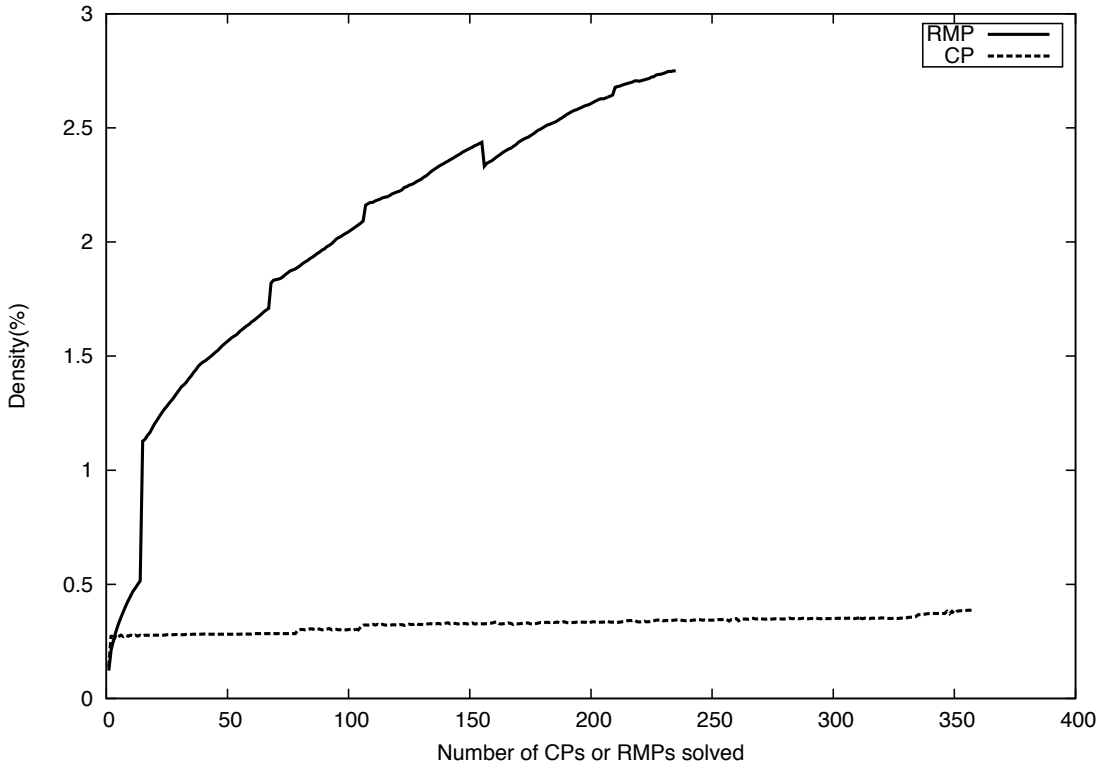

Figure 5: Density (%) of CP vs RMP constraint matrix

average size of these partitions is 20% of the number of set partitioning constraints in the original master problem. Unlike in [6], the partitions are not reaggregated during the optimization. The small size of the partitions is essentially a result of the high quality of the columns generated during the first phases using the compatibility matrix M^2.

5.3 Dual solutions

The dual solutions, provided by ARMP and completed by CP, reduce the number of CGen iterations by a factor of four in IDCA compared to SCG. Figure 6 illustrates the stability of the dual solutions on a medium instance (of 800 tasks) by representing the *dual distance* between two successive iterations, t and $t+1$, given by $\|\alpha_t - \alpha_{t+1}\|_\infty$. The dual solutions obtained in the intermediate minor iterations are not considered since they are not used by SP to generate columns. We also ignore, for the sake of clarity, the CGen iterations where the primal solution of RMP/ARMP is not feasible. The optimal solution is reached at CGen iteration 67 in IDCA while SCG reaches it at iteration 220. The distances between the dual solutions of IDCA are approximately a multiple of 50000, i.e., the cost of a driver. This value is reached before and after saving a driver, after disaggregating the ARMP, and after a phase change. Around iteration 15, the IDCA distance becomes small because it coincides with the end of the first phase. In contrast, the SCG distance decreases slowly to 50000 before vanishing on a long tail. This is due to the RMP degeneracy.

Figure 7 reports the improvements in the objective value from one iteration to the next one (between two successive calls to SP). The improvements are measured by the formula $\frac{z^t - z^{t+1}}{z^t - z^\star}$ where z^t is the ARMP objective function value at iteration t and $z^\star$ is the optimal value. The IDCA improvements are quite important from the first iterations, while the SCG improvements become important after the SCG dual distance reaches 50000. SCG requires stability of the dual variables in order to significantly improve the objective value. This stability is quickly reached by IDCA because of the row and column aggregation. Aggregating the rows of RMP reduces the number of nonzero variables in the basis and therefore reduces

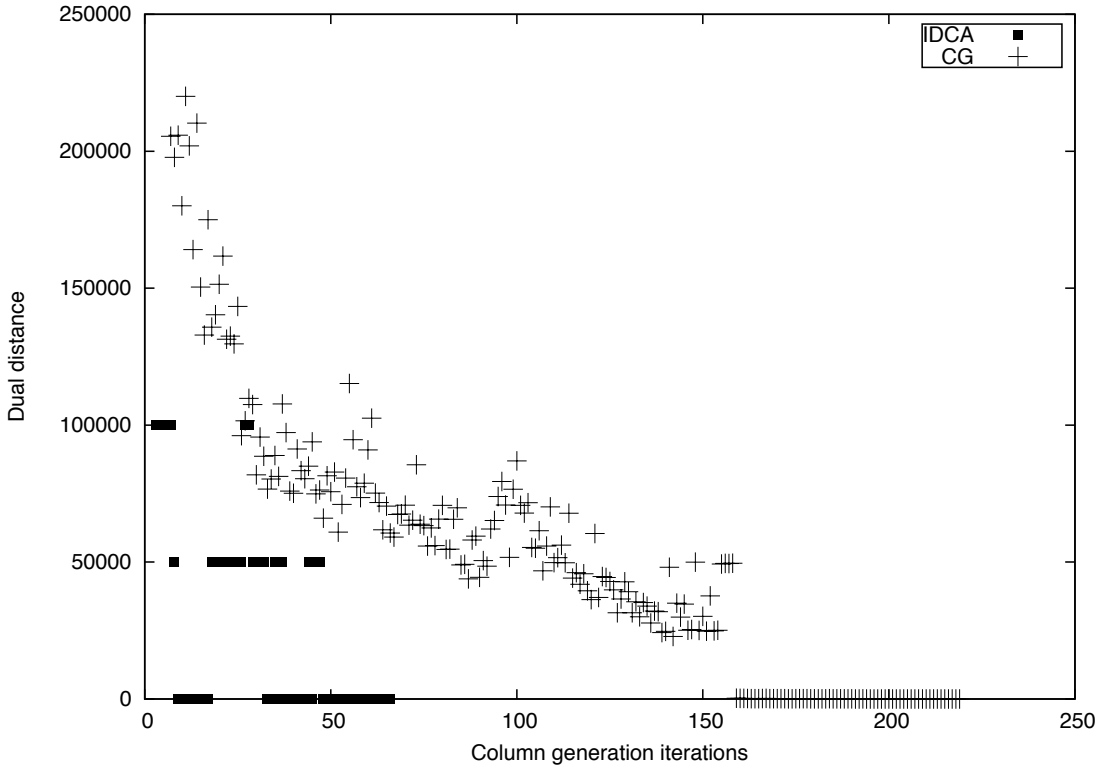

Figure 6: Distance between two successive dual solutions

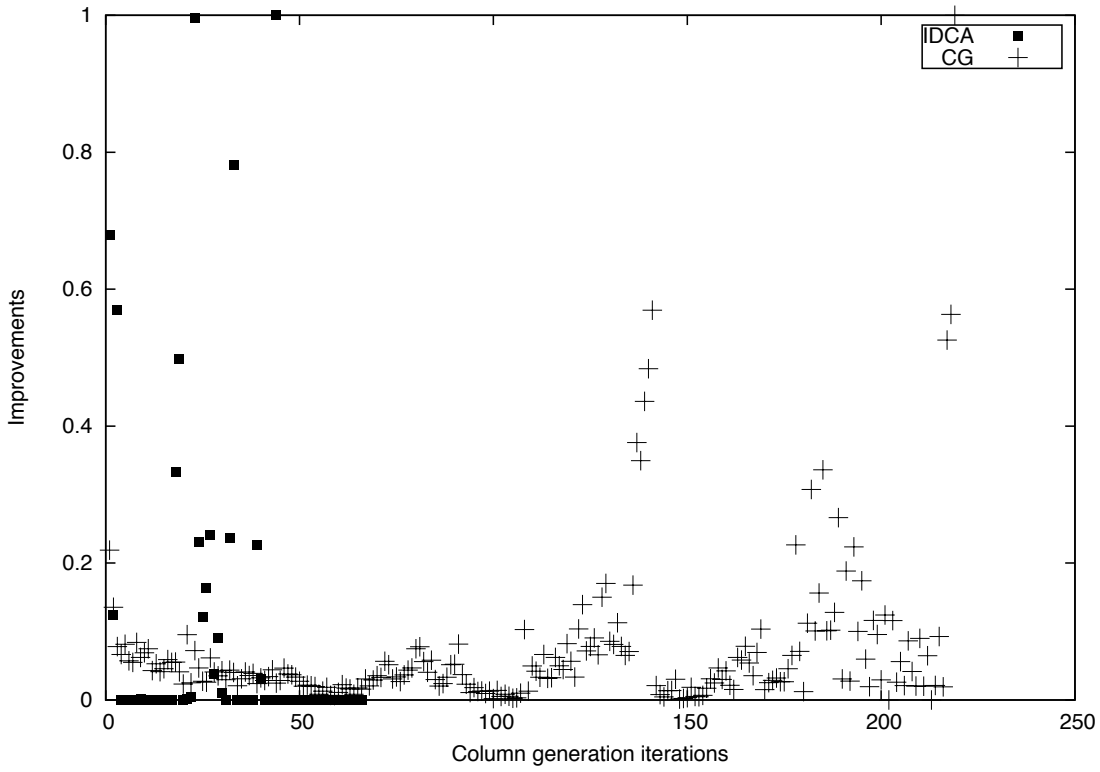

Figure 7: Improvements in the objective value

degeneracy. The column aggregation improves the ARMP dual solutions without increasing the problem size, as discussed below.

The CP is built from the incompatible columns by maximizing their minimum reduced cost μ. If $\mu < 0$ then a compatible combination of incompatible columns with negative reduced costs is detected. In a minor iteration, the positive variables in a CP solution are combined to form one aggregated compatible variable to add to ARMP. The main objective of this strategy is not to improve the objective value but to improve the aggregated dual-solution quality of ARMP without increasing the size of the current partition. In fact, for a medium VCSP instance of 800 tasks, of the improvements in the ARMP objective value that exceed 0.1 according to the formula given above, only 18% are made by the aggregated columns during the minor iterations. Figure 8 shows the proportion of positive aggregated variables in ARMP (number of positive aggregated variables/number of positive variables) for the 800-task instance. This proportion increases after we add aggregated columns to ARMP (after a minor iteration) and decreases quickly after we add nonaggregated columns. Because the aggregated columns are generally dense, they are hard to combine with other columns in the basis of ARMP, and therefore they are more likely to leave the basis. The aggregated columns could be part of the optimal basis of ARMP without corrupting its extremal property.

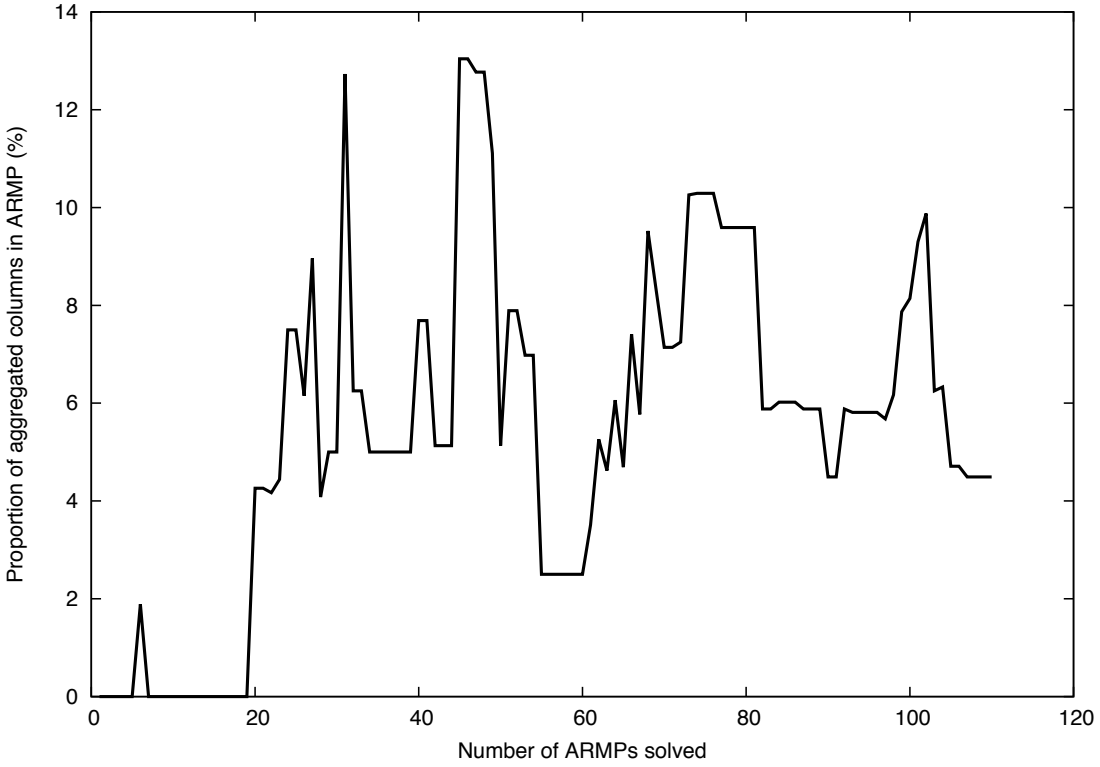

Figure 8: Proportion of positive aggregated variables in ARMP

If the minimum reduced cost μ is nonnegative then all the incompatible columns have nonnegative reduced costs, and ARMP is equivalent to an RMP containing the compatible and incompatible columns. When $\mu \geq 0$, the dual solution given by CP is used to generate new columns from SP. Since CP is easy to solve, an important number of columns can be considered to improve the dual solutions (giving a better approximation of the dual polyhedron). For IDCA, the average number of columns of CP and ARMP for the largest instances is 6 times more than the average number of columns of RMP (in CG), while the average time to solve ARMP and CP is 10 times less than the time to solve RMP.

The time spent in the SP is also affected by the compatibility matrix and the quality of the dual solutions. In SCG where RMP is degenerate, a greater computational effort, due mainly to the dominance procedure used when solving the subproblem by the labeling algorithm [11], is required to find interesting columns

because of the poor quality of the dual solutions. In IDCA, this effort is lessened since the dual solutions are good, and hence the dominance is good. The dual solutions and the compatibility matrix of IDCA reduce the time spent in SP by a factor of 11 compared to SCG and reduce the number of iterations by a factor of 4 on average.

5.4 Acceleration strategy

In this section, we present an improved version of IDCA that we refer to as I^{2}DCA. Reducing the number of columns of CP significantly accelerates IDCA. We rank the incompatible columns in increasing order of the ratio $\frac{\tilde{c}_j}{k_j}$. Those with unattractive ratios are removed. In I^{2}DCA, the computational times, reported in Tables 5.2 and 5.3, are reduced on average by 21% compared to IDCA. The time is cut, on average, by 50% for CP (up to 66% for the largest instances) by reducing the number of columns in CP by an average of 39% (up to 57% for the largest instances) without increasing its density. The partition size at the end of the optimization process increases by 6% because fewer columns are in CP. The dual-solution quality is lower because the dual polyhedron approximation is not as good as in IDCA (which retains more columns). The ARMP therefore needs more disaggregations to prove optimality. The number of iterations increases by 6% and SP requires 9% more time for the largest instances because of the loss in the quality of the dual solutions and the increase in the size of the partition.

Table 5.4: Numerical results for I^{2}DCA

Algorithm	I^{2}DCA					
Instance	400	600	800	1200	1600	2000
Time	1.13	8.07	9.77	52.40	180.17	402.90
CgItr	36	70	59	81	125	136
MinorItr	25	80	61	158	244	277
RmpT	0.13	1.70	1.37	11.37	45.07	89.80
CpT	0.20	1.70	2.00	13.63	43.43	76.00
SpT	0.63	4.10	5.60	24.13	80.87	216.10
RmpC	204	643	779	1971	3969	5498
CpC	978	1936	2703	4516	7067	8292
PartSize	66	140	145	272	411	505

In addition to removing columns with uninteresting ratios, the formula above eliminates dense columns with small negative reduced costs. Such columns, because of their density, contribute to many CP constraints, hence increasing the probability of degeneracy or leading to small steps. This strategy preserves the columns with an interesting ratio of the reduced cost and the density, which is affected by the incompatibility degree. Columns with few incompatibilities form small combinations when added to ARMP, do not disaggregate the partition too much, which means less degeneracy in ARMP, and significantly improve the objective value when the reduced cost is interesting.

6 Concluding remarks

This paper presents an improved CGen framework based on the improved primal simplex and CGen decompositions. In this framework, we solve a nondegenerate RP and a CP to find efficient descent directions leading to primal basic solutions with improved objective value in a CGen context. In this context, the role of the CP is twofold: 1) it recovers "interior" full-dimensional dual vector from an aggregated dual vector, more stable than dual vectors of DCA/MPDCA and SCG, to generate good columns, and 2) it selects columns with a good potential to improve the objective value of the RP from among those generated by the subproblem. We specialize this improved CGen to the SPP to obtain an improved dynamic constraint aggregation. We show that this version is a nice generalization of our previous work on DCA and MPDCA. We also discuss a specialized compatibility matrix for some vehicle and crew scheduling problems that favors good columns in the early phases of the solution process and reduce the number of nonzero elements in the complementary constraint matrix.

This new framework opens up a new research question about how to improve column generation more intelligently, finding an option for solving the RMP between extreme methods such as the primal simplex method and more central methods such as interior point methods.

References

[1] Barnhart, C., E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance (1998). Branch-and-Price: Column Generation for Solving Huge Integer Programs. Operations Research 46, 316–329.

[2] Desrochers, M., J. Desrosiers, and F. Soumis (1992). A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. Operations Research 40, 342–354.

[3] Desrochers, M. and F. Soumis (1989). A Column Generation Approach to the Urban Transit Crew Scheduling Problem. Transportation Science 23, 1–13.

[4] Desrosiers, J., J.B. Gauthier, and M.E. Lübbecke (2014). Row-reduced Column Generation for Degenerate Master Problems. European Journal of Operational Research 236(2), 453–460.

[5] Elhallaoui, I., D. Villeneuve, F. Soumis, and G. Desaulniers (2005). Dynamic Aggregation of Set Partitioning Constraints in Column Generation. Operations Research 53, 632–645.

[6] Elhallaoui, I., A. Metrane, F. Soumis, and G. Desaulniers (2010). Multi-phase Dynamic Constraint Aggregation for Set Partitioning Type Problems. Mathematical Programming Series A 123(2), 345–370.

[7] Elhallaoui, I., A. Metrane, G. Desaulniers, and F. Soumis (2011). An Improved Primal Simplex Algorithm for Degenerate Linear Problems. INFORMS Journal on Computing 23(4), 569–577.

[8] Gamache, M., F. Soumis, G. Marquis, and J. Desrosiers (1999). A Column Generation Approach for Large Scale Aircrew Rostering Problems. Operations Research 47, 247–263.

[9] Gauthier, J.B., J. Desrosiers, and M.E. Lübbecke (2014). About the minimum mean cycle-canceling algorithm. Discrete Applied Mathematics.

[10] Haase, K., G. Desaulniers, and J. Desrosiers (2001). Simultaneous Vehicle and Crew Scheduling in Urban Mass Transit Systems. Transportation Science 35, 286–303.

[11] Irnich, S., G. Desaulniers (2005). Shortest path problems with resource constraints. G. Desaulniers, J. Desrosiers, M. M. Solomon, eds. Column Generation. Springer, New York, 33–65.

[12] Lübbecke, M. and J. Desrosiers (2005). Selected Topics in Column Generation. Operations Research 53(6), 1007–1023.

[13] Metrane, A., F. Soumis, and I. Elhallaoui (2010). Column Generation Decomposition with the Degenerate Constraints in the Subproblem. European Journal of Operational Research 207(1), 37–44.

[14] Vanderbeck F. (2005). Implementing Mixed Integer Column Generation. In: Desaulniers G., Desrosiers J., Solomon M., editors. Column Generation. Springer US, pp. 33–58.