Les Cahiers du GERAD ISSN: 0711–2440

Dynamic constraint aggregation for solving very large-scale airline crew pairing problems

G. Desaulniers, F. Lessard,
M. Saddoune, F. Soumis

G–2020–21

April 2020

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

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2020
– Bibliothèque et Archives Canada, 2020

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

Legal deposit – Bibliothèque et Archives nationales du Québec, 2020
– Library and Archives Canada, 2020

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 aggregation for solving very large-scale airline crew pairing problems

Guy Desaulniers [a,b]

François Lessard [a,b]

Mohammed Saddoune [b,c]

François Soumis [a,b]

[a] *GERAD, Montréal (Québec), Canada, H3T 2A7*

[b] *Department of Mathematics and Industrial Engineering, Polytechnique Montréal (Québec) Canada, H3C 3A7*

[c] *University of Hassan II, FST of Mohammedia, Department of Computer Science, 20650 Morocco*

guy.desaulniers@polymtl.ca
francois.lessard@gerad.ca
mohammed.saddoune@polymtl.ca
francois.soumis@polymtl.ca

April 2020
Les Cahiers du GERAD
G–2020–21

Abstract: The monthly crew pairing problem (CPP) consists of determining a least-cost set of feasible crew pairings (sequences of flights starting and ending at a crew base) such that each flight is covered once and side constraints are satisfied. This problem has been widely studied but most works have tackled daily or weekly CPP instances with up to 3500 flights. Only a few papers have addressed monthly instances with up to 14000 flights. In this paper, we propose an effective algorithm for solving very large-scale CPP instances. This algorithm combines, among others, column generation (CG) with dynamic contraint aggregation (DCA) that can efficiently exploit the CG master problem degeneracy. When embedded in a rolling-horizon (RH) procedure, DCA allows to consider wider time windows in RH and yields better solutions. Our computational results show, first, the potential gains that can be obtained by using wider time windows and, second, the very good performance of the proposed algorithm when compared to a standard CG/RH algorithm for solving an industrial monthly CPP instance with 46588 flights.

Keywords: Airline crew pairing problem, large scale instances, column generation, dynamic constraint aggregation

---

Acknowledgments: We would like to thank the personnel of Ad Opt for providing the industrial datasets used in our computational experiments, discussing the proposed algorithms, and validating the computed solutions. We are also very grateful to Kronos Inc. and the Natural Sciences and Engineering Research Council of Canada for their financial support (grant RDC477127–14).

Conflict of interest: G. Desaulniers and F. Soumis have received a research grant from the company Kronos who owned Ad Opt when this research project was conducted. Ad Opt is now part of the company IBS Software.

1 Introduction

When planning the operations of an airline, the crew scheduling phase consists of determining the work schedules of a set of crew members in order to operate a given one-week or one-month flight schedule at minimum cost. Given its complexity, this problem is usually decomposed in two problems that are solved sequentially: the crew pairing problem (CPP) and the crew rostering problem (CRP). The CPP aims at building least-cost crew pairings such that each flight is covered by at least one pairing and some side constraints are satisfied. A *pairing* is a sequence of one or more duties separated by rest periods which starts and ends at the same crew base (an airport where crew members are assigned). A *duty* is a sequence of flights separated by connections which form a working day. It can contain one or several *deadheads*, i.e., flights where the crew travels as passengers to be repositioned. Duties and pairings are subject to numerous feasibility rules imposed by the labour agreement and the aviation authorities. Given a set of pairings, the CRP consists of constructing work schedules for a given set of crew members assigned to different crew bases such that each pairing is covered by a sufficient number of crew members and other side constraints are met. A work schedule is composed of a set of pairings interspersed with days off and pre-assigned activities such as training periods and vacations. It must also respect a variety of feasibility rules. A typical objective function for the CRP is to maximize the satisfaction of the preferences expressed by the employees. In this paper, we focus on the CPP. For more information of the CRP, see, e.g., the recent surveys of Kasirzadeh et al. (2017) and Deveci and Demirel (2018).

The CPP is generally modeled as a set partitioning or set covering problem and, due to high degeneracy, the difficulty to solve it increases rapidly with the number of flights to cover. For this reason, the monthly problem has been traditionally solved using a three-phase approach (see, e.g., Gopalakrishnan and Johnson, 2005). In the first phase, a cyclic daily problem that considers a flight schedule for a typical day is solved assuming that this schedule repeats every day. Unrolling the computed solution over a week while filtering out the copies of the pairings that are infeasible, the second phase solves a cyclic weekly problem defined for a typical weekly flight schedule that minimizes the total cost and the number of changes compared to the daily solution. The third phase is similar as it unrolls the weekly solution over a month and solves an acyclic monthly problem to cover all dated flights of a one-month schedule, while minimizing the total cost and the changes to the pairings selected in the second phase.

The CPP has been extensively studied in the literature. Early works (some of the latest being Hoffman and Padberg, 1993, Chu et al., 1997, and Klabjan et al., 2001) have focused on generating a subset of feasible pairings a priori and solving the resulting set partitioning/covering using a commercial mixed-integer program (MIP) solver. Such solution methods offer no guarantee on the solution quality, especially for large instances.

To achieve high quality solutions while keeping the computational times under control for medium-sized CPP instances, a variety of column generation (CG) algorithms have been devised, including those of Lavoie et al. (1988), Desaulniers et al. (1997), Vance et al. (1997), Barnhart and Shenoi (1998), AhmadBeygi et al. (2009), Dück et al. (2011), Muter et al. (2013), Saddoune et al. (2013), Zeren and Özkol (2016), Quesnel et al. (2020a,b), and Parmentier and Meunier (2020). CG is an iterative algorithm that solves the linear relaxation of the set-partitioning/covering model, called the master problem. At each iteration, it first solves the master problem restricted to a small subset of variables to derive a primal and a dual solution before solving a pricing problem that searches for negative reduced cost pairing variables. If some are found, they are added to the restricted master problem; otherwise, the algorithm stops. For the CPP, the pricing problem is separable per crew base and day of the horizon and corresponds to a shortest path problem with resource constraints (see Irnich and Desaulniers, 2005).

To derive integer solutions, CG is usually embedded in a (possibly heuristic) branch-and-bound or branch-and-cut framework, yielding a so-called branch-and-price(-and-cut) algorithm (see Desaulniers et al., 1998, Barnhart et al., 1998, Desaulniers et al., 2005). Alternatively, Parmentier and Meunier

(2020) used CG to derive a lower bound on the optimal value, before applying a pairing enumeration procedure and solving, using a MIP solver, a restricted set-partitioning model that ensures finding an optimal solution. On the other hand, Tahir et al. (2019a,b) have devised a primal algorithm that combines CG and the integral simplex using decomposition algorithm (ISUD) of Zaghrouti et al. (2014). ISUD is used to solve the restricted master problem taken into account the integrality requirements. The algorithm is qualified as primal because it yields a sequence of improved integer solutions.

Most of these papers have addressed daily or weekly CPP instances with less than 3500 flights. Only the following four papers have tackled monthly instances. Saddoune et al. (2013) applied a rolling-horizon (RH) technique where overlapping CPP subproblems restricted to a 3-day time window each are sequentially solved by CG. They solved instances with up to 7500 flights in approximately 8 hours of computational time. Quesnel et al. (2020a,b) developed similar algorithms to solve instances of CPP variants considering crew preferences and language constraints. Zeren and Özkol (2016) used a very aggressive connection fixing technique to obtain integer solutions for instances containing between 12000 and 14000 flights. Given the size of these instances, their computational times can be deemed as very fast (less than 40 minutes) but their instances have characteristics that substantially simplify them, namely, there is a single crew base and the selected pairings contain only around 1.2 duties and 2.6 flights on average.

The main contribution of this paper is to propose an effective CG/DCA-based algorithm for solving very large-scale CPP instances. The most important ingredient of this algorithm is dynamic constraint aggregation (DCA), a technique introduced by Elhallaoui et al. (2005) that exploits degeneracy to reduce the number of constraints to consider in the CG master problem. The algorithm also includes a RH scheme and the search for integer solutions using a MIP solver. All these ingredients are not novel but, to our knowledge, they have not been put together yet for tackling large-scale CPP instances. A lighter algorithm version, which excludes RH among others, is also developed to tackle academic monthly instances (those used by Saddoune et al., 2013) and show the potential gains that can be produced by solving CPP instances without applying RH. Finally, to illustrate the power of the full-fledged algorithm, we compare its performance to that of a standard CG/RH algorithm for solving an industrial monthly CPP instance with 46588 flights.

This paper is organized as follows. Section 2 defines the CPP and formulates it as a set-partitioning problem. Section 3 describes the proposed algorithms. Section 4 reports the results of our computational experiments on academic and industrial CPP instances. Finally, conclusions are drawn in Section 5.

2 Problem statement and mathematical model

The CPP is separable by aircraft type since, for safety reasons, the crew members are trained for a specific aircraft type (or, for the flight attendants, a family of similar aircraft). Given a set of flights F to operate over a planning horizon, the CPP consists of building a set of feasible pairings such that each flight in F is actively covered by a single pairing, a set S of soft side constraints is satisfied, and the total operating cost is minimized. A flight is said to be actively covered by a pairing if it contains this flight and its assigned crew is not deadheading on it. A pairing is deemed feasible if it satisfies a number of regulations such as a maximum duration, a minimum connection time between consecutive flights, a minimum rest time between consecutive duties, a maximum number of duties and, for each duty, a minimum and maximum working time, a maximum duration and a maximum number of landings. Set S can typically include base constraints and pairing distribution constraints. For each crew base, the former impose the assignment of pairings that induce a minimum total working time that depends on the number of available crew members at this base. The latter constraints often model some crew member preferences for short or long pairings, or for pairings with specific features.

In practice, the cost of a pairing is complex as it is a function of its total duration, the duration of each of its duties, the flying time and deadhead time in each duty, and it includes a guaranteed minimum flying time per duty (see, e.g., Quesnel et al., 2017). For the academic instances (see

Saddoune et al., 2013, Mercier et al., 2005), the cost of a pairing is given by an approximate function that considers a cost for each deadhead that depends on its duration, a guaranteed minimum flying time per duty and some penalties for short/long connection times between two consecutive flights and short/long rest times between two consecutive duties. In this case, the total cost of assigning an active crew to each flight is seen as a constant that does not have to be considered in the model. The CPP cost structure also involves penalties associated with the soft side constraints.

The CPP is mathematically formulated using the sets F and S defined above and the following notation. Let P be the set of all feasible pairings. For each pairing $p \in P$, let c_p be its cost, a_f^p be a binary parameter that takes value 1 if pairing p actively covers flight $f \in F$, and e_s^p be the contribution of pairing p to side constraint $s \in S$. Moreover, define a binary variable x_p that is equal to 1 if p is selected in the solution and 0 otherwise. Next, for each side constraint $s \in S$, let b_s be its right-hand side which, typically, corresponds to a minimum number to attain. Furthermore, define a slack variable y_s that is penalized at a rate of γ_s for each unit short of b_s.

Given this notation, the CPP can be formulated as the following set partitioning problem with side constraints:

$$\text{Minimize} \qquad \sum_{p \in P} c_p x_p + \sum_{s \in S} \gamma_s y_s \tag{1}$$

$$\text{subject to:} \qquad \sum_{p \in P} a_f^p x_p = 1, \qquad \forall f \in F \tag{2}$$

$$\sum_{p \in P} e_s^p x_p + y_s = b_s, \qquad \forall s \in S \tag{3}$$

$$x_p \in \{0,1\}, \qquad \forall p \in P \tag{4}$$

$$y_s \geq 0, \qquad \forall s \in S. \tag{5}$$

The objective function (1) minimizes the total cost of the selected pairings plus the sum of the side constraints penalties. Constraints (2) ensure that each flight is actively assigned to exactly one pairing. Side constraints such as base constraints are expressed by (3). Note that they could also represent soft less-than-or-equal constraints with surplus variables instead of slack variables. Finally, constraints (4) and (5) restrict the domain of the decision variables.

In practice, model (1)–(5) contains a huge number of x_p variables that cannot be enumerated a priori. To overcome this drawback, it is solved by CG combined with DCA as discussed in the next section.

3 Solution algorithms

This section presents the different CG-based algorithms that we used for solving the CPP. They involve various components (CG, DCA, branching/variable fixing, MIP-based search for integer solutions, and rolling horizon) that have already been proposed in the literature. Given that describing all these components in detail would be too lengthy, we only summarize the main ideas for some of them and refer the readers to appropriate publications.

We first describe these components in Subsections 3.1 to 3.5, before presenting the algorithms used for our tests in Subsection 3.6.

3.1 Column generation

To solve the linear relaxation of model (1)–(5) which is called the master problem, we can apply a CG algorithm (see, e.g., Desrosiers and Lübbecke, 2005, Lübbecke and Desrosiers, 2005). CG is an iterative algorithm that alternates between a restricted master problem (RMP) and one or several

pricing problems. For the CPP, the RMP is given by the master problem restricted to a subset of the x_p variables which is updated at each iteration. It is solved by the primal simplex algorithm to provide a primal and a dual solution. Given this dual solution, the pricing problems aim at finding negative reduced cost x_p variables. They are formulated as shortest path problems with resource constraints as described below and solved by dynamic programming. If no negative reduced cost variables are found, then the CG process stops because the current primal solution is optimal for the linear relaxation. Otherwise, the columns found (or a subset of them) are added to the RMP before starting a new iteration.

There is a pricing problem for each crew base and each day of the planning horizon that allows the generation of pairings assigned to this base and starting on the corresponding day. Each of them can be modeled as a shortest path problem with resource constraints defined on an acyclic time-space network, where a node represents an airport and a time and an arc a movement in space or time (e.g., a flight or a connection). For the academic instances, we use the same network structure as in Saddoune et al. (2013), where each flight in F that can be covered by a pairing starting on the day associated with the pricing problem is represented by an arc. This structure ensures that all source-to-sink paths correspond to a pairing starting and ending at the same crew base that respects the maximum duration of a pairing as well as some other pairing feasibility rules. The other rules are modeled using resource constraints (see, e.g., Desrosiers et al., 1995, and Irnich and Desaulniers, 2005). A resource is a quantity that varies along a path and is restricted to take a value within a so-called resource window at each visited node. Resource examples are the duration of a duty and the number of duties in a pairing. For these instances, five resources are required to ensure pairing feasibility. The pricing problems are solved exactly by a labeling algorithm (see Irnich and Desaulniers, 2005).

For the industrial instances which involve a much larger number of pairing feasibility rules as well as a complex pairing cost, we could not resort to the same network structure because it would require more than forty resources. Consequently, to reduce the number of resources and speed up the labeling algorithm, a set of duties is generated a priori by a heuristic devised by our industrial partner and a duty-based network structure underlies each pricing problem, where each duty in this set that can be covered by a pairing starting on the day associated with the pricing problem is represented by an arc. With this approach, duty feasibility is treated during network construction and only around twenty-five resources are needed to correctly model pairing feasibility and compute the cost of a pairing. Given that this number of resources remains high, a heuristic labeling algorithm which considers a label dominance rule involving only five main resources is applied. Hence, there is no guarantee that a lower bound has been found when stopping CG.

3.2 Dynamic constraint aggregation

Due to the set partitioning constraints (2), the linear relaxation of model (1)–(5) induces high degeneracy. To overcome this difficulty, we combine CG with the DCA algorithm of Elhallaoui et al. (2005, 2010). DCA reduces degeneracy by using an aggregated restricted master problem (ARMP) that is obtained from a partition Q of the set of flights F into disjoint clusters. For each cluster in Q, the ARMP contains a single set partitioning constraint (2). A variable (pairing) x_p is said to be compatible with respect to partition Q if the set of flights covered by the corresponding pairing is the union of some clusters in Q. Otherwise, this variable is declared incompatible. Each variable has a degree of incompatibility that is equal to the minimum number of additional clusters needed in the partition to make the variable compatible. Beside the y_s variables, the ARMP only includes the compatible x_p variables. Note that DCA is qualified as dynamic because partition Q and the ensuing ARMP evolve during the solution process.

Figure 1 illustrates the DCA algorithm. It begins by choosing an initial partition Q (Step 1) derived from an initial solution (computed, for instance, using a heuristic) by setting one cluster for each pairing in this solution. Then, it performs two types of iterations: the minor iterations are embedded within the major ones. A minor iteration consists of three steps. In Step 2, the current ARMP is solved using a linear programming solver to compute a primal solution and a dual solution that contains a dual value

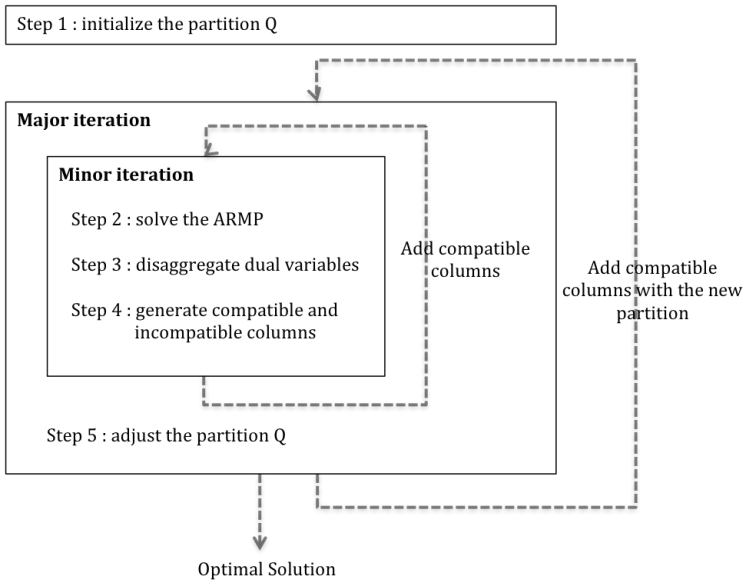

Figure 1: DCA process

for each aggregated constraint. In Step 3, these dual values are disaggregated using a series of shortest path problems to obtain one dual value for each set-partitioning constraint (2) of the original master problem. Given these dual values, the pricing problems are solved in Step 4 to try to generate negative reduced cost columns that may be compatible or not. The following three outcomes are possible, where $\bar{z}^C$ (resp. $\bar{z}^I$) is equal to the reduced cost of the least-reduced-cost compatible (resp., incompatible) column found or 0 if no such columns are found. First, when no negative reduced cost columns are found, the algorithm stops with an optimal solution to the master problem. Second, when compatible columns are found and $z^I \geq \rho z^C$ for a predefined parameter $\rho > 0$ (e.g., $\rho = 2$), the compatible columns are added to the ARMP before starting a new minor iteration. Third, when incompatible columns are generated and $z^I < \rho z^C$ (i.e., the incompatible columns are sufficiently attractive compared to the compatible ones), partition Q is adjusted to allow some incompatible columns to become compatible in Step 5 before starting a new major iteration. See Elhallaoui et al. (2005) for more details.

DCA takes advantage of degeneracy by defining an ARMP whose size is inversely proportional to the level of degeneracy in the current solution. Furthermore, the computational results in Elhallaoui et al. (2005) have shown that DCA yields optimal solutions to the master problem which have less fractional-valued variables than the solutions obtained with a standard CG algorithm. To further encourage this behavior, Elhallaoui et al. (2010, 2008) have introduced two other versions of the DCA algorithm, named multi-phase DCA and bi-DCA (or MPDCA and BDCA), that favor the generation of columns that are compatible or have a small degree of incompatibility with respect to the current partition Q.

3.2.1 Multi-phase and bi-dynamic constraint aggregation

Before summarizing MPDCA and BDCA, let us discuss how the degree of incompatibility of a pairing is computed while solving a pricing problem. To facilitate the understanding, consider the auxiliary network presented in Figure 2 for a pricing problem containing nine tasks (flights or duties) $T1$ to $T9$ represented by horizontal arcs and grouped into four clusters $C1$ to $C4$. The network also contains start-of-pairing, end-of-pairing and inter-tasks arcs. With each arc is associated an incompatibility number 0, 1 or 2 (only the numbers 1 and 2 are shown in the figure) which is equal to the number of the arc nodes that is within a cluster, i.e., not the first or last node of a cluster. For instance, in Figure 2, the incompatibility number of the arc linking the source node to the tail node of arc $T2$ is equal to 1 because a pairing beginning with task $T2$ can be compatible only if cluster $C1$ is split in two;

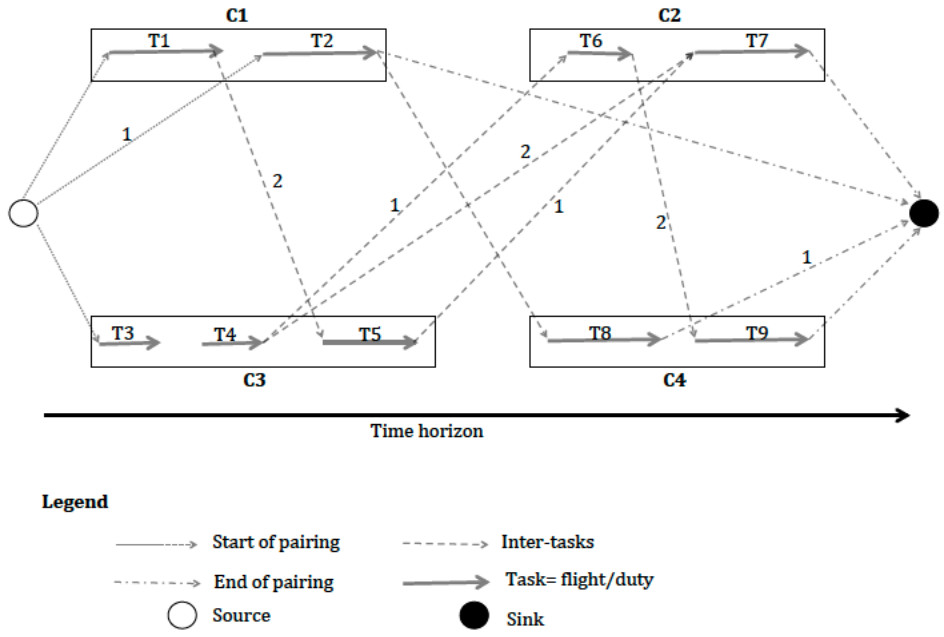

Figure 2: Auxiliary network to illustrate the degree of incompatibility

that of the arc linking the head node of arc $T6$ to the tail node of arc $T9$ is equal to 2 because a pairing performing tasks $T6$ and $T9$ consecutively can be compatible only if clusters $C2$ and $C4$ are broken up. Given these incompatibility numbers, the degree of incompatibility of a pairing is computed as the sum of the incompatibility numbers of the arcs contained in the path representing it.

The structure of the time-space network underlying a pricing problem differs from the auxiliary network structure. Indeed, the tasks are not linked by inter-tasks arcs but rather by sequences of arcs, including waiting arcs at an airport. Consequently, it is not possible to associate an incompatibility number with each arc. Instead, a label (a vector of information representing a partial path) in the labeling algorithm contains an additional component that indicates the last flight covered on the path. When extending a label to cover a new flight, it then becomes possible to compute the incompatibility number between the last flight and the new flight. Note also that, in the duty-based networks used for the industrial instances, the same flight may appear in several tasks, thus, in several task arcs. In this case, the flights of a task may belong to more than one cluster. Therefore, an incompatibility number is also associated with each task (duty) arc.

MPDCA (Elhallaoui et al., 2010) proceeds through a predetermined sequence of phases, typically, phases $k = 0, 1, 2, \infty$. In phase k, only pairings with a degree of incompatibility not exceeding k can be generated by the pricing problems. To impose this constraint in the pricing problem, an additional incompatibility degree resource is considered. Each phase stops when no negative reduced cost columns are found.

Developed after MPDCA, BDCA (Elhallaoui et al., 2008) is an extension of MPDCA that temporarily modifies the pricing problem networks to avoid generating columns that would break some selected clusters. At each iteration, a certain percentage of the clusters are selected based on the dual values associated with the flights they contain and all arcs that yield an incompatibility with these clusters are removed from the networks. Then, the labeling algorithm is invoked to solve the pricing problems. If negative reduced cost columns are generated, the algorithm continues with them. Otherwise, all removed arcs are put back in the networks and the pricing problems are solved again.

3.2.2 Improved dynamic constraint aggregation

Bouarab et al. (2017) developed an improved version of DCA that is denoted IDCA (for Improved DCA). It speeds up the solution of the master problem by computing disaggregated dual values that are better (more stable) than those obtained from shortest path problems as in Elhallaoui et al. (2005), yielding a reduced number of CG iterations. As proposed in Elhallaoui et al. (2011), the dual values are computed by solving a linear program which aims at finding compatible convex combinations of incompatible columns (previously generated) that have a negative reduced cost. When such combinations are identified, a single variable representing each combination is added to the ARMP without modifying the current partition Q. When CG stops, all combination variables that take a positive value in the optimal solution are replaced in the ARMP by the columns composing these combinations (which may require adjusting the current partition) while the other combination variables are discarded from the ARMP. This facilitates branching on the pairing variables.

3.3 Branching and variable fixing

To derive integer solutions, the CG/DCA algorithm is embedded in a branch-and-bound heuristic, that is, the lower bounds are computed by CG at each node of the search tree. To rapidly reduce the size of the problem and avoid large search tree, variable fixing is applied in a diving fashion, mostly at the top of the tree. More precisely, after solving a linear relaxation, certain decisions are imposed without the possibility to backtrack by creating a single child node. Two types of decisions are considered. First, all x_p variables that take a fractional value greater than a predetermined threshold (for our tests, 0.7 for the weekly instances and 0.6 for the monthly instances) are fixed to 1. Second, if no such variables exist, we compute, for each pair of flights $f_1, f_2 \in F$, the flow w_{f_1,f_2} of pairings that cover f_2 immediately after f_1, i.e., $w_{f_1,f_2} = \sum_{p \in P_{f_1,f_2}} x_p$, where P_{f_1,f_2} is the set of these pairings. Then, for each pair of flights f_1, f_2 for which w_{f_1,f_2} is fractional and larger than the given threshold, an inter-task constraint forcing both flights to be covered consecutively by the same pairing is imposed. These inter-task constraints are handled in the pricing problem labeling algorithm as described in Irnich and Desaulniers (2005).

Finally, if no such decisions can be imposed, dichotomic branching is performed by selecting the pair of flights $f_1, f_2 \in F$ with the largest fractional w_{f_1,f_2} value. On one branch, f_1 and f_2 are forced to be covered consecutively by the same pairing; on the other, they must be covered by two different pairings or by the same pairing but not consecutively. These inter-task decisions are also treated as specified in Irnich and Desaulniers (2005).

Note that variable fixing can be performed after imposing branching decisions and that new columns are generated after fixing variables or branching.

3.4 Search for integer solutions using a mixed integer program

To increase the chances of finding good integer solutions during the solution process, a MIP solver can be called to solve the current RMP after adding integrality requirements on the x_p variables. In fact, it is called just before branching for the first time and also after solving every linear relaxation if the computed solution has less than a pre-specified number of fractional-valued variables (for our tests, 350 for the weekly instances and 550 for the monthly instances). In both cases, a maximum time limit is imposed to the MIP solver.

To speed up the MIP solution process, subset row inequalities (see Jepsen et al., 2008) that are violated by the current RMP solution are added to the RMP before calling the MIP solver. These cuts are removed once the MIP is solved.

3.5 Rolling horizon

A RH procedure (see, e.g., Saddoune at al., 2013) solves a sequence of subproblems to gradually fix parts of the solution. More precisely, it first divides the time horizon into a sequence of overlapping

time windows (in chronological order). For example, a 30-day month can be divided into ten time windows of four days with a one-day overlap: the first time window would span days 1 to 4, the second days 4 to 7, the third days 7 to 10, and so on. Then, for each time window, it solves the subproblem restricted to the coverage of the flights contained in this window, considering that the solutions computed for the previous windows are fixed up to the beginning of the current window. Note that, for each window, the flights to be operated on the first few days after the window are also considered in the pricing problems to allow the completion of pairings after the time window. Clearly, RH can significantly reduce computational times but its greedy nature may yield solutions that are clearly suboptimal. In particular, it is not well suited to handle global side contraints that involve decisions made over long time periods such as the base constraints.

3.6 Algorithm variants

As detailed in the next section, our computational experiments were performed on three categories of CPP instances, namely, seven academic monthly instances, three industrial cyclic weekly instances, and one industrial monthly instance. Given that the characteristics (size, cyclic/acyclic, number of resources considered in the pricing problems) of the instances vary from one category to another, different algorithms were compared for each category.

For the academic monthly instances (moderate size, acyclic, a few resources), the algorithms are:

CG-RH-A: A standard (non-commercialized) CG algorithm embedded in a RH procedure (3-day time windows with a 1.5-day overlap);

DCA: A non-improved DCA algorithm without the MIP-search strategy.

For the industrial cyclic weekly instances (large size, cyclic, many resources), the algorithms are:

CG-RH-IW: A standard (non-commercialized) CG algorithm that uses a heuristic dominance rule in the pricing algorithm and is embedded in a RH procedure (2-day time windows with a 1-day overlap). Some flights may remain uncovered in the first time window as not all flights can be covered using pairings starting on the first day. To cover these flights and ensure that a cyclic solution is computed, the RH procedure contains seven windows. The last one is composed of the last and first days of the week and includes final conditions to ensure continuity of the first-window pairings that were spanning over the first and second days;

IDCA-MIP: An IDCA algorithm, including the MIP-search strategy, that uses a heuristic dominance rule in the pricing algorithm.

For the industrial monthly instance (very large size, acyclic, many resources), the algorithms are:

CG-RH-IM: A standard (non-commercialized) CG algorithm that uses a heuristic dominance rule in the pricing algorithm and is embedded in a RH procedure (2-day time windows with a 1-day overlap);

IDCA-MIP-RH: An IDCA algorithm, including the MIP-search strategy, that uses a heuristic dominance rule in the pricing algorithm and is embedded in a RH procedure (7-day time windows with a 2-day overlap).

All these algorithms include the branching/variable fixing strategy described in Subsection 3.3. All DCA/IDCA-based algorithms include the MPDCA/BDCA techniques. Furthermore, to speed up the computational times of the IDCA-MIP and IDCA-MIP-RH algorithms, only the phases $k = 0, 1, 2$ of the MPDCA are performed and IDCA might be stopped before reaching optimality in every node of the search tree. In fact, IDCA is stopped if the size of the partition Q exceeds the size of the initial partition Q by 1500 (resp. 2000) for the weekly (monthly) instances. This ensures that the ARMP remains sufficiently aggregated and usually yields linear relaxation solutions that contain less fractional-valued variables.

4 Computational experiments

To test the performance of the various algorithms, we conducted computational experiments on academic monthly instances and industrial weekly and monthly instances. The academic instances are used to analyze the impact of using RH on the quality of the solutions. To the best of our knowledge, the industrial instances are the largest ones for which computational results are reported in a published scientific paper. Our tests were performed on a Linux PC machine (equipped with an Intel Xeon processor clocked at 3.3 GHz), using a single core and thread. Our implementation is coded in C/C++ and uses the GENCOL column generation library (version 4.5) and the solvers of CPLEX 12.4 for tackling the (A)RMPs and the MIPs.

Before presenting the results of our computational tests in Sections 4.2 and 4.3 for the academic and industrial instances, respectively, we describe these instances.

4.1 Test instances

Tables 1 and 2 provides certain statistics on the academic and industrial instances, respectively. Taken from Saddoune et al. (2013), the seven academic instances A-1 to A-7 were derived from a one-month flight schedule operated by a major North American airline, which was divided by aircraft type. There are 3 crew bases and between 1011 and 7527 flights per instance. Pairing feasibility is restricted by a limited number of rules (the most common ones) including a maximum pairing duration of four days, and the pairing cost function is approximated as stipulated in Saddoune et al. (2013). There are no side constraints (3). As stated in Section 3.1, the pricing problems for these instances require five resources and can be solved to optimality. Therefore, valid lower bounds can be computed and optimality gaps can be estimated. For these reasons, the academic instances are useful to calibrate and assess the performance of the DCA algorithm and, indirectly, the IDCA algorithm that is used for solving the industrial instances.

Table 1: Academic monthly instances from Saddoune et al. (2013)

Instance	Flights	Airports	Bases
A-1	1011	26	3
A-2	1463	35	3
A-3	1793	41	3
A-4	5466	49	3
A-5	5639	34	3
A-6	5755	52	3
A-7	7527	54	3

Table 2: Industrial weekly (W) and monthly (M) instances

Instance	Flights	Airports	Bases
IW-5	9367	157	7
IW-6	9261	160	7
IW-7	8711	161	7
IM-5	46588	161	7

Our industrial partner Ad Opt gave us large-scale industrial instances from an airline (see Table 2). There are three weekly instances, denoted IW-i, $i \in \{5, 6, 7\}$ (for May, June and July), and one monthly instance IM-5. The weekly instances involve 7 crew bases and between 8711 and 9367 flights (i.e., regular flights for the corresponding month that operate each week). The monthly instance includes 46588 flights, 7 bases and a total of 119 side constraints.

4.2 Results for the academic instances

The seven academic monthly instances are used to assess the impact on the total solution cost of using a RH procedure for solving the CPP. Given that these instances are of moderate size, we can compute

using algorithm DCA a valid lower bound on the optimal value and bound the optimality gap of the computed integer solution for each instance. All instances were first solved using algorithm CG-RH-A (the same as in Saddoune at al., 2013). Then, they were solved a second time using DCA that used the computed CG-RH-A solutions to define the DCA initial partitions Q.

Table 3 reports the results obtained by DCA. For each instance, it gives the total computational time (excluding the time to compute the initial clusters), the gap in percentage between the cost of the computed solution and the lower bound at the root node of the search tree, the cost saving in percentage with respect to the CG-RH-A solution, the number of pairings, duties and deadheads (DH) in the computed solution and the number of deadheads saved with respect to the CG-RH-A solution. The last line provides some averages.

Table 3: DCA results for the academic instances

Instance	Time	Gap (%)	Cost saving (%)	No. pairings	No. duties	No. DH	DH saving (%)
A-1	0h17	0.15	4.52	128	304	9	77.5
A-2	0h25	0.29	1.08	301	419	0	100.0
A-3	0h28	0.01	3.70	240	582	0	100.0
A-4	4h38	0.36	0.37	1066	1577	27	15.6
A-5	0h56	0.00	0.38	1446	2890	58	18.3
A-6	3h57	0.13	1.94	964	2200	47	27.7
A-7	2h21	0.37	1.36	1578	3544	123	12.8
Average		0.19	1.90				25.6

From these results, we observe that all instances have been solved in a reasonable computational time of no more than 4 hours and 38 minutes. This is quite impressive given the size of the instances and the quality of the computed solutions which exhibit an average optimality gap of less than 0.19%. Compared to the CG-RH-A solutions, DCA produces solutions with an average cost saving of 1.9%, showing the negative impact of using a RH procedure. This is substantial knowing that a cost reduction of 1% typically represents tens of millions dollars per year for a medium-sized airline. These savings are partially due to an average reduction of 25.6% on the number of deadheads. From the other statistics, we can deduce that, in the computed solutions, there are on average 2.03 duties and 5.71 flights per pairing, highlighting the complexity of these instances.

Note that we do not report the computational times of CG-RH-A as this algorithm was run on a different computer and served to compute the initial partitions of DCA. Nevertheless, as a rough comparison, CG-RH-A seems to have been faster than DCA for the three smallest instances and slower or much slower for the others.

4.3 Results for the industrial instances

The three industrial cyclic weekly instances were solved using the algorithms CG-RH-IW and IDCA-MIP, where the IDCA initial partition Q was constructed from the computed CG-RH-IW solution for each instance. Here again, these tests allow to evaluate the impact of using a RH procedure with short time windows. The results obtained by IDCA-MIP are reported in Table 4 which presents the same statistics as in Table 3, except that the savings are computed with respect to the solutions obtained with CG-RH-IW and no gaps are specified because no valid lower bounds are provided by IDCA-MIP.

Table 4: IDCA-MIP results for the industrial weekly instances

Instance	Time	Cost saving (%)	No. pairings	No. duties	No. DH	DH saving (%)
IW-5	3h36	1.31	1135	2693	26	42.2
IW-6	4h38	3.45	916	2548	133	21.8
IW-7	4h39	17.09	1009	2616	109	-23.9
Average		7.28				13.4

These results show again that tackling the problem at once instead of solving a sequence of subproblems restricted by the RH time windows can yield substantial savings, with an average of 7.28%. It should be noted that a large part of this average gain comes from instance IW-7 for which a saving of 17.09% was achieved. In fact, this huge saving is attributed to a large reduction of the violation of the soft global constraints and does not directly incur a so important reduction of the operational costs. Nevertheless, reducing the global constraint penalties in the CPP always has a positive impact either on the satisfaction of the crew members or on the CRP cost. Note that this reduction was achieved in part by increasing considerably the number of deadheads used in the solution. On the other hand, for the other two instances, the large decrease in the number of deadheads certainly brings a significant contribution to the cost savings.

The results also indicate that IDCA-MIP can solve these instances in less than 5 hours. This is certainly due to the CG stopping criterion described at the end of Subsection 3.6 that aims at limiting the number of fractional-valued variables in the linear relaxation solutions. In fact, for these three instances, a maximum of 2187 fractional-valued variables appeared in the root node linear relaxation solutions, yielding a maximum of 255 branch-and-bound nodes in the search trees.

Our final goal was to compare the proposed IDCA-MIP-RH algorithm to the standard CG-RH-IM algorithm for solving a very large industrial monthly instances. This time, the initial partition Q used by IDCA-MIP-RH was not computed from the CG-RH-IM solution but rather by from the solution of a weekly instance solved using IDCA. The results of these experiments for both algorithms are reported in Table 5. The savings obtained by IDCA-MIP-RH are with respect to the solution computed by CG-RH-IM. Note that the IDCA-RH-RH computational time includes in this case the time to compute the IDCA initial partition Q.

For this industrial instance, IDCA-MIP-RH clearly outperforms CG-RH-IM with a computational time reduction of around 33% and a cost saving of 23.85%. The time reduction is due to the usage of IDCA that exploits degeneracy and control the number of fractional-valued variables in the linear relaxation solutions. The huge cost saving is induced by a large reduction of the violations of the global constraints (the penalties are reduced by 77%), while the pairing costs are decreased by 9.3%. These results are not surprising because using 2-day time windows in the RH procedure of CG-RH-IM is clearly too myopic to generate a good solution and to handle global constraints properly. Using 7-day time windows as in IDCA-MIP-RH is much better.

Table 5: Results for the industrial monthly instance IM-5

Algorithm	Time	Cost saving (%)	No. pairings	No. duties	No. DH	DH saving (%)
CG-RH-IM	92h03	-	5928	17284	1224	-
IDCA-MIP-RH	61h20	23.85	5201	16473	740	39.5

One can also observe that the IDCA-MIP-RH solution contains close to 5% less duties and 40% less deadheads than the CG-RH-IM solution. This indicates that the duties are much more productive and, thus, yield lower operational costs. Finally, notice that, in the IDCA-MIP-RH solution, there is an average of 3.2 duties and 8.9 flights per pairing, which is larger than the average of 1.2 duties and 2.6 flights per pairing encountered in the instances considered by Zeren and Özkol (2016). Solving this monthly instance with 46588 flights is, thus, highly challenging!

5 Conclusion

In this paper, we have introduced CG/DCA-based algorithms for solving large-scale CPP instances. Our computational results show that these algorithms outperform standard CG/RH algorithms, clearly highlighting that DCA or its improved version IDCA is efficient at solving large instances and considering much wider time windows in a RH procedure. Solving weekly instances at once or considering wider RH time windows allows a better handling of the global constraints and yields substantial operational cost savings.

For our computational results, we derived the initial partition of (I)DCA from a heuristic solution that was computed using a CG/RH or DCA algorithm. The computational times needed to find these solutions are important and have exceeded, sometimes largely, the times of proposed algorithms. Clearly, as a follow-up project to this work, the development of a new methodology to determine a good initial partition seems interesting to reduce the overall computational times. In this respect, one idea that has been explored by Yaakoubi (2019) in parallel to the writing of this paper is to design a machine learning tool for predicting the connections between consecutive flights and designing initial clusters for (I)DCA. Further investigations are, however, still required to ensure that all this methodology is sufficiently robust for application in the industry.

References

AhmadBeygi S, Cohn A, Weir M (2009). An integer programming approach to generating airline crew pairings. Computers & Operations Research, 36(4): 1284–1298.

Baldacci R, Christofides N, Mingozzi A (2008). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. Mathematical Programming 115(2): 351–385.

Barnhart C, Shenoi R (1998). An approximate model and solution approach for the long-haul crew pairing problem. Transportation Science, 32(3): 221-231.

Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH (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, 262: 835–850.

Chu HD, Gelman E, Johnson EL (1997). Solving large scale crew scheduling problems. European Journal of Operational Research, 97: 260–268.

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

Desaulniers G, Desrosiers J, Ioachim I, Solomon MM, Soumis F, Villeneuve D (1998). A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In Crainic TG and Laporte G, Fleet Management and Logistics, Norwell, MA, pp 57–93.

Desaulniers G, Desrosiers J, Solomon MM, Eds. (2005). Column generation, Springer, New York.

Desrosiers J, Dumas Y, Solomon MM, Soumis F (1995). Time constrained routing and scheduling. In Ball M et al., Network Routing, Handbooks in Operations Research and Management Science, volume 8, Elsevier, Amsterdam, pp 35–139.

Desrosiers J, Lübbecke M (2005). A primer in column generation. In Desaulniers G, Desrosiers J, Solomon MM, Column Generation, chap. 1, Springer, New York, pp 1–32.

Deveci M, Demirel NÇ (2018). A survey of the literature on airline crew scheduling. Engineering Applications of Artificial Intelligence, 74: 54-69

Dück V, Wesselmann F, Suhl L (2011). Implementing a branch and price and cut method for the airline crew pairing optimization problem. Public Transport, 3(1): 43–64.

Elhallaoui I, Desaulniers G, Metrane A, Soumis F (2008). Bi-dynamic constraint aggregation and subproblem reduction. Computers and Operations Research, 35(5): 1713–1724.

Elhallaoui 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.

Elhallaoui I, Metrane A, Soumis F, Desaulniers G (2010). Multiphase dynamic constraint aggregation for set partitioning type problems. Mathematical Programming, 123(2): 345–370.

Elhallaoui I, Villeneuve D, Soumis F, Desaulniers G (2005). Dynamic aggregation of set partitioning constraints in column generation. Operations Research, 53: 632–645.

Gopalakrishnan, B, Johnson, EL (2005). Airline crew scheduling: State-of-the-art. Annals of Operations Research, 140: 305–337.

Hoffman KL, Padberg M (1993). Solving airline crew scheduling problems by branch-and-cut. Management Science, 39: 657–682.

Irnich S, Desaulniers G (2005). Shortest path problems with resource constraints. In: Desaulniers, G., Desrosiers, J., Solomon, M.M. (Eds.), Column Generation. Springer, New York, pp. 33–65.

Jepsen M, Petersen B, Spoorendonk S, Pisinger D (2008). A non-robust branch-and-cut-and-price algorithm for the vehicle routing problem with time windows. Operations Research, 43: 297–325.

Kasirzadeh A, Saddoune M, Soumis F (2017). Airline crew scheduling: models, algorithms, and data sets. EURO Journal on Transportation and Logistics 6(2): 111–137.

Klabjan D, Johnson E, Nemhauser G, Gelman E, Ramaswamy, S (2001). Solving large airline crew scheduling problems: Random pairing generation and strong branching. Computational Optimization and Applications, 20: 73–91.

Lavoie S, Minoux M, Odier E (1988). A new approach for crew pairing problems by column generation with an application to air transportation. European Journal of Operational Research, 35(1): 45–58.

Lübbecke ME, Desrosiers J (2005). Selected topics in column generation. Operations Research 53(6): 1007–1023.

Mercier A, Cordeau JF, Soumis F (2005). A computational study of Benders decomposition for the integrated aircraft routing and crew scheduling problem. Computers and Operations Research, 32: 1451–1476.

Muter I, Ilker Birbil S, Bulbul K, Sahin G, Yenigun H, Tas D and Tuzun D (2013). Solving a robust airline crew pairing problem with column generation. Computers & Operations Research, 40(3): 815–830.

Parmentier A, Meunier F (2020). Aircraft routing and crew pairing: Updated algorithms at Air France. Omega, forthcoming.

Quesnel F, Desaulniers G and Soumis F (2020a). A branch-and-price heuristic for the crew pairing problem with language constraints. European Journal of Operational Research, 283(3): 1040–1054.

Quesnel F, Desaulniers G and Soumis F (2020b). Improving air crew rostering by considering crew preferences in the crew pairing problem. Transportation Science, 54(1): 97–114.

Saddoune M, Desaulniers G, Elhallaoui I, Soumis F (2011). Integrated airline crew scheduling: A bi-dynamic constraint aggregation method using neighborhoods. European Journal of Operational Research, 212(3): 445–454.

Saddoune M, Desaulniers G, Elhallaoui I, Soumis F (2012). Integrated airline crew pairing and crew assignment by dynamic constraint aggregation. Transportation Science, 46(1): 39–55.

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

Tahir A, Desaulniers G, Elhallaoui I (2019a). Integral column generation. EURO Journal on Transportation and Logistics, 8(3): 713–744.

Tahir A, Desaulniers G, Elhallaoui I (2019b). Integral column generation for set partitioning problems with side constraints, Technical report, Les Cahiers du GERAD G-2019-85, HEC Montréal, 25 pages.

Vance PH, Barnhart C, Johnson EL, Nemhauser GL (1997). Airline crew scheduling: A new formulation and decomposition algorithm. Operations Research, 45: 188–200.

Yaakoubi, Y (2019). Combiner intelligence artificielle et programmation mathématique pour la planification des horaires des équipages en transport aérien. PhD dissertation, Polytechnique Montréal, Montréal, Canada.

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