Les Cahiers du GERAD ISSN: 0711–2440

A Graph Coloring Model for a Feasibility Problem in Crew Scheduling

M. Gamache, A. Hertz,
J. Ouellet

G–2005–34

April 2005

A Graph Coloring Model for a Feasibility Problem in Crew Scheduling

Michel Gamache

GERAD and Département des génies civil, géologique et des mines
École Polytechnique de Montréal
C.P. 6079, Succ. Centre-ville
Montréal (Québec) Canada H3C 3A7
michel.gamache@polymtl.ca

Alain Hertz, Jérôme Ouellet

GERAD and Département de mathématiques et de génie industriel
École Polytechnique de Montréal
C.P. 6079, Succ. Centre-ville
Montréal (Québec) Canada H3C 3A7
{alain.hertz;jerome.ouellet}@polymtl.ca

April 2005

Les Cahiers du GERAD

G–2005–34

Abstract

We consider a crew scheduling problem with preferential bidding in the airline industry. We propose a new methodology based on a graph coloring model and a tabu search algorithm for determining if the problem contains at least one feasible solution. We then show how to combine the proposed approach with a heuristic sequential scheduling method that uses column generation and branch-and-bound techniques.

Key Words: Graph coloring, crew scheduling, preferential bidding, tabu search.

Résumé

Nous considérons un problème de confection d'horaires d'équipages en transport aérien, dans lequel on tient compte des préférences des employés. Nous proposons une nouvelle méthodologie basée sur un modèle de coloration de graphe et sur un algorithme de type Recherche avec tabous, le but étant de déterminer si le problème considéré contient au moins une solution qui respecte toutes les contraintes. Nous montrons ensuite comment combiner l'approche proposée avec une heuristique séquentielle de confection d'horaires qui utilise la génération de colonnes et une technique de séparation-évaluation (branch-and-bound).

1 Introduction

In the airline industry, the problem of constructing monthly work schedules consists of designing a monthly schedule for each employee while taking into account their preferences and a list of preassigned activities such as vacations and training periods. More precisely, a *schedule* is a sequence of pairings, pre-assignments, and rest periods. A *pairing* is a sequence of flights that starts from a city, called *base*, and ends at this same city. A schedule is *feasible* if it respects constraints imposed by security rules of the airline industry, collective agreements between the employees and the company, and pre-assignment requirements. The *crew scheduling problem* is to determine a feasible schedule for each employee such that the selected schedules cover all the pairings planned during the month.

Each employee draws up a list of preferences that is used for calculating a value reflecting the interest (or the aversion) of the employee for each pairing and rest period. The *score* of a schedule for an employee is computed as the sum of the values assigned to the pairings and rest periods in his schedule. Two modes can be used to build the schedules. In the first mode, the objective consists of maximizing the global satisfaction of the employees, that is, the overall sum of the selected schedule scores. This mode is called *rostering* and is mainly used in large European airline companies such as Air France (Giafferri et al. 1982, Gontier 1985, Gamache et al. 1998a), Alitalia (Nicoletti 1975, Marchettini 1980, Sarra 1988, Federici et Paschina 1990), Lufthansa (Glanert 1984), SwissAir (Tingley 1979), Air New-Zealand (Ryan 1992) and El-Al Israel Airline (Mayer 1980). The second mode, called *preferential bidding*, is similar to the first one except that the assignment of tasks (pairings and rest periods) according to preferences is done on the basis of seniority rules. More precisely, this mode requires that no employee can get a better score if this new score imposes a strict decrease in the score of a more senior employee. Preferential bidding is most often used in large North-American airlines.

The crew scheduling problem with preferential bidding (PBS) is quite new and thus less known than the rostering problem. There are only few papers in the literature that deal with this problem. Companies such as Quantas (Moore et al., 1978), CP Air (Byrne, 1988), Midwest Express Airlines, Inc. (Schmidt, 1994), Air Canada (Gamache et al., 1998) have developed their solution method. Based on Sherali and Soyser's approach on preemptive multi-objective programming (1983), one can formulate the PBS as an integer linear program. However, as observed in (Gamache et al., 1998), such a model necessitates huge numbers that are much too large for any computer. Because of the complexity of the problem, most of the methods presented in the literature are based on greedy heuristics.

The current most efficient heuristic method for the PBS is a sequential algorithm based on column generation and branch and bound (Gamache et al, 1998, Achour et al. 2004). It solves a sequence of mixed linear programs, one for each employee, from the most senior to the most junior employee. It may happen that the method fixes a schedule for an employee so that the more junior employees cannot cover the residual pairings. A backtracking

process is therefore used to repair bad decisions. The method of Gamache et al. (1998) is described in the next section. In order to avoid the above mentioned backtracks, which may be very time consuming, we propose to add a feasibility test that determines if it is possible to assign a feasible schedule to each employee while covering all pairings. This feasibility problem is explained with more details in Section 3, and modeled as a graph coloring problem in Section 4. A solution method for the proposed graph coloring problem is described in Section 5, and some computational experiments are reported in Section 6. Final remarks and conclusions are drawn in Section 7.

2 Heuristic sequential algorithms for the PBS

Let P denote a set of pairings that have to be covered, and let $E = \{1, \cdots, m\}$ be the set of employees labeled from the most senior to the most junior. The heuristic sequential method for the PBS proposed by Gamache et al. (1998) constructs the schedules sequentially, one after the other, from the most senior employee to the most junior one. In such a sequential algorithm, a problem is defined for each employee $i \in E$. It consists of constructing the best possible schedule for employee k, without changing the schedules already chosen for the more senior employees $1, \cdots, k-1$, and with the constraint that it should be possible to assign feasible schedules to the more junior employees $k+1, \cdots, m$ so that all pairings in P are covered.

The assignment of employees to pairings must satisfy qualification requirements imposed by the airline company. Let Q be the set of qualifications, and define δ_{iq} equal 1 if employee i has qualification q, 0 otherwise. Assume that a feasible schedule has already been assigned to employees $1, \cdots, k-1$. We will use the following notations:

- N_{pq}^k represents the number of employees with qualification q still required by pairing p.
- b_p^k represents the number of employees still required by pairing p ;
- S_i^k is the set of all remaining feasible schedules for employee i ;

Also, let a_{pj} equal 1 if pairing p is part of schedule j, 0 otherwise, and let c_{ij} denote the score of schedule j for employee i. Finally, let x_{ij} be a binary variable which equals 1 if schedule $j \in S_i^k$ is chosen for employee i, and 0 otherwise. The problem of finding a feasible schedule for employee k so that the schedules of more junior employees can cover the residual pairings can be formulated as an integer linear program (IP_k) as follows.

$$
(IP_k)\begin{cases}
\quad Max\ Z_{IP_k} = \displaystyle\sum_{j\in S_k^k} c_{kj}\ x_{kj} & & & & \\
\text{subject to} & & & & \\
\displaystyle\sum_{i=k}^{m}\sum_{j\in S_i^k} a_{pj}\ \delta_{iq}\ x_{ij} & \geq & N_{pq}^k & \forall p\in P\ \ \forall q\in Q & (1)\\
\displaystyle\sum_{i=k}^{m}\sum_{j\in S_i^k} a_{pj}\ x_{ij} & = & b_p^k & \forall p\in P & (2)\\
\displaystyle\sum_{j\in S_i^k} x_{ij} & = & 1 & i=k,\ldots,m & (3)\\
x_{ij} & \in & \{0,1\} & i=k,\ldots,m,\ \ \forall j\in S_i^k & (4)
\end{cases}
$$

Qualification requirements are imposed by constraints (1). The constraint set (2) ensures that the set of selected schedules covers all the pairings with the appropriate number of employees. Constraints (3) guarantee that a schedule is built for each employee. Binary requirements on the x_{ij} variables are given by (4).

Let $x_1^*,\cdots,x_{k-1}^*$ denote feasible schedules assigned to employees $1,\cdots,k-1$. The integer linear program (IP_k) resulting from these assignments will be denoted $IP(x_1^*,\cdots,x_{k-1}^*)$. An ideal sequential algorithm for the PBS, called PSB-IP is given in Figure 1. It obviously produces an optimal solution to PBS in the case where no employee has two schedules with the same score.

Procedure PSB-IP

for $k=1,\cdots,m$ do
  determine an optimal solution s to $IP(x_1^*,\cdots,x_{k-1}^*)$;
  set x_k^* equal to the schedule assigned to k in s;

Figure 1: An sequential heuristic algorithm for the PBS

The solution of (IP_k) may however prove very long. Moreover, in the optimal solution of (IP_k), the schedules for employees $k+1,\cdots m$ are useless for the subsequent iterations since the preferences of these employees have not been taken into account. To reduce the solution time, Gamache et al. (1998) propose to solve the following mixed integer linear program (MIP_k), where integrality constraints are imposed only for the variables associated with employee k.

$$
(MIP_k)\begin{cases}
Max\ Z_{MIP_k} = \displaystyle\sum_{j\in S_k^k} c_{kj}\ x_{kj} & & & & \\
\text{subject to} & & & & \\
\displaystyle\sum_{i=k}^{m}\sum_{j\in S_i^k} a_{pj}\ \delta_{iq}\ x_{ij} & \geq & N_{pq}^k & \forall p\in P\ \ \forall q\in Q & (1)\\
\displaystyle\sum_{i=k}^{m}\sum_{j\in S_i^k} a_{pj}\ x_{ij} & = & b_p^k & \forall p\in P & (2)\\
\displaystyle\sum_{j\in S_i^k} x_{ij} & = & 1 & i=k,\ldots,m & (3)\\
x_{kj} & \in & \{0,1\} & \forall j\in S_k^k. & (4')\\
x_{ij} & \geq & 0 & i=k+1,\ldots,m,\ \ \forall j\in S_i^k. & (4'')
\end{cases}
$$

Gamache et al. (1998) propose to solve (MIP_k) by combining a column generation algorithm with a branch-and-bound technique. Column generation is used to solve the linear program obtained by relaxing the integrality constraints in (MIP_k), while branch-and-bound is used to assign an integer solution to employee k.

It often happens that (MIP_k) and (IP_k) have the same optimal value. This situation is however not always true and it may therefore happen that a (MIP_k) has no feasible solution. In such a case, one must backtrack and find an optimal solution to (IP_{k-1}), if possible; otherwise, this backtracking process is repeated until a feasible solution is obtained for an (IP_ℓ) with $\ell < k$. The process, called PSB-MIP, is summarized in Figure 2, where $MIP(x_1^*,\cdots,x_{k-1}^*)$ denotes the (MIP_k) resulting from the assignment of schedules $x_1^*,\cdots,x_{k-1}^*$ to employees $1,\cdots,k-1$. In this process, it is assumed that (IP_1) has a feasible solution.

Procedure PSB-MIP

(0) set k:=1;

(1) if $MIP(x_1^*,\cdots,x_{k-1}^*)$ has no feasible solution then go to step (2)
else determine an optimal solution s to $MIP(x_1^*,\cdots,x_{k-1}^*)$ and go to (3)

(2) repeat $k := k-1$ until $IP(x_1^*,\cdots,x_{k-1}^*)$ has a feasible solution;
determine an optimal solution s to $IP(x_1^*,\cdots,x_{k-1}^*)$ and go to (3)

(3) let x_k^* by the schedule assigned to k in s; set $k := k+1$, and go to (1);

Figure 2: The sequential algorithm for the PBS proposed by Gamache et al. (1998)

We illustrate the backtracking process with the following simple example where $m = 5$ employees must cover a set $P = \{A, B, C, D, E, F, G\}$ of pairings. We suppose that all

pairings require exactly one employee, except G that requires two employees. The periods associated with the pairings are represented in Figure 3. Each pairing has a number of flight credits corresponding to the number of units on the horizontal axis. For example, D has 2 flight credits while E has 5 flight credits. We assume that the workload of each employee must be of at least 6 flight credits, but not more than 10 flight credits, and that pairing D is pre-assigned to employee 3.

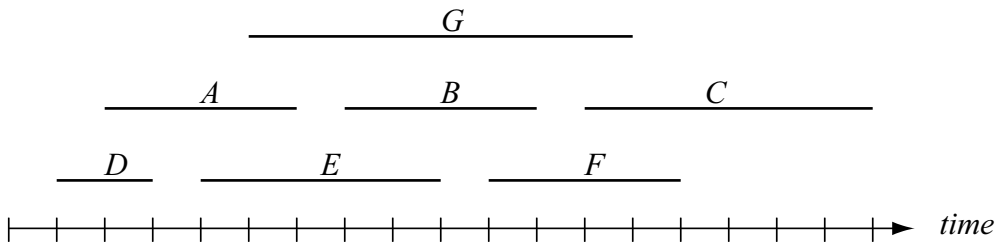

Figure 3: Seven pairings

The sets of feasible schedules at iteration 1 are

- $S_i^1 = \{\emptyset, A, B, C, E, F, G, A \rightarrow B, A \rightarrow C, A \rightarrow F, B \rightarrow C, E \rightarrow F\}$ $(i = 1, 2, 4, 5)$
- $S_3^1 = \{D, D \rightarrow B, D \rightarrow C, D \rightarrow E, D \rightarrow F, D \rightarrow G\}$

where $p_1 \rightarrow p_2$ denotes the schedule made of pairing p_1 followed by pairing p_2.

Consider the set $Q = \{\alpha, \beta, \gamma, \delta\}$ of qualifications, and assume that

- pairing A requires one employee with qualification α,
- pairings B and C both require one employee with qualification β,
- pairings E and F both require one employee with qualification γ, and
- pairing G requires one employee with qualification δ.

Suppose also that employees 1 and 4 have qualification α, employees 1, 4 and 5 have qualification β, employees 3 and 5 have qualification γ, and employees 1 and 3 have qualification δ, while employee 2 has no qualification. Finally, assume that employee 1 has given a score 10 to G, 4 to A and B, 2 to C, and 0 to D, E and F.

The optimal solution to MIP_1 is the following one with value 10:

- $x_{1G} = x_{2G} = 1$,
- $x_{3D \rightarrow E} = x_{3D \rightarrow F} = x_{4A \rightarrow B} = x_{4A \rightarrow C} = x_{5B \rightarrow C} = x_{5E \rightarrow F} = 0.5$, and
- $x_{ij} = 0$ otherwise.

However, (IP_1) has no feasible solution of score 10. Indeed, if one gives pairing G to 1, then 1 cannot do anything else, which means that employees 3, 4 and 5 must cover pairings $A, \cdots, F$ (while employee 2 can only be assigned to pairing G). But this means that

(1) employee 4 must do pairing A since he is the last one with qualification α. Hence, he cannot do both B and C because of the limit of 10 flight credits in his workload. As a consequence, employee 5 must do B or/and C (since both pairing require an employee with qualification β).

(2) employee 3 cannot do both E and F (because he must do D and cannot cumulate more than 10 flight credits). As a consequence, employee 5 must do E or/and F (since both pairing require an employee with qualification γ).

(1) and (2) are contradictory since there is no feasible schedule for employee 5 that contains a pairing from $\{B, C\}$ and a pairing from $\{E, F\}$.

As a conclusion, the optimal solution of (MIP_1) assigns pairing G to employee 1, and the consequence is that (IP_2) has no feasible solution. This is not detected immediately since (MIP_2) has a feasible solution $x_{2G} = 1$, $x_{3D\to E} = x_{3D\to F} = x_{4A\to B} = x_{4A\to C} = x_{5B\to C} = x_{5E\to F} = 0.5$, and $x_{ij} = 0$ otherwise. By assigning pairing G to employee 2, one gets (MIP_3) which has no feasible solution. The backtracking process must then first detect that (IP_2) has no feasible solution, and finally produce an optimal solution to (IP_1) which consists in assigning schedule $A \to B$ to 1, schedule G to 2, schedule $D \to G$ to 3, schedule C to 4 and schedule $E \to F$ to 5. This solution has value 8 which is strictly smaller that the optimal value of (MIP_1).

Instead of solving (MIP_k) or (IP_k), one can generate a schedule for employee k without taking care of the more junior employees. This corresponds to solving the following problem, called $(RCSP_k)$, which is a resource constrained shortest path problem (this algorithm is used by Gamache et al., 1998, for solving the subproblems in the column generation approach).

$$(RCSP_k) \begin{cases} Max\ Z_{RCSP_k} = \displaystyle\sum_{j \in S_k^k} c_{kj}\ x_{kj} & \\ \text{subject to} & \\ \displaystyle\sum_{j \in S_k^k} x_{kj} = 1 & \\ x_{kj} \in \{0,1\} & \forall j \in S_k^k. \end{cases}$$

It often happens in practice that (IP_k), (MIP_k) and $(RCSP_k)$ have the same optimal value, especially for small values of k. The reason is that the number of more junior employees is large enough to ensure the satisfaction of constraints (1) to (3) of (IP_k).

Jeandroz (1999) and El Idrissi (2002) have proposed to use counters to detect when the number of more junior employees becomes critical (i.e., there is a non-negligible probability that the schedule obtained by solving $RCSP_k$ involves an infeasible solution for IP_k). The counters measure the number of required employees and qualifications at each time period, and these demands are compared to the current offers. When the demands are too close to the offers, the method switches to (MIP_k) instead of $(RCSP_k)$. Details are given in Jeandroz (1999). Procedure PBS-MIP-RCSP in Figure 4 presents a possible way to integrate counters in a solution scheme, where $RCSP(x_1^*, \cdots, x_{k-1}^*)$ denotes the $(RCSP_k)$ resulting from the assignment of schedules $x_1^*, \cdots, x_{k-1}^*$ to employees $1, \cdots, k-1$. An alternative, proposed by Jeandroz (1999) and El Idrissi (2002), is to never turn back to Step 2 as soon as Step 3 is visited.

Procedure PSB-MIP-RCSP

(0) set k:=1;

(1) use counters to determine whether to go to step (2) or (3);

(2) if $RCSP(x_1^*, \cdots, x_{k-1}^*)$ has no feasible solution then go to step (4)
else determine an optimal solution s to $RCSP(x_1^*, \cdots, x_{k-1}^*)$ and go to (5);

(3) if $MIP(x_1^*, \cdots, x_{k-1}^*)$ has no feasible solution then go to step (4)
else determine an optimal solution s to $MIP(x_1^*, \cdots, x_{k-1}^*)$ and go to (5);

(4) repeat $k := k-1$ until $IP(x_1^*, \cdots, x_{k-1}^*)$ has a feasible solution;
determine an optimal solution s to $IP(x_1^*, \cdots, x_{k-1}^*)$ and go to (5);

(5) let x_k^* be the schedule assigned to k in s; set $k := k+1$, and go to (1);

Figure 4: A sequential algorithm that combines (IP_k), (MIP_k), $(RCSP_k)$,

3 A feasibility problem

As already noticed, it may happen that (IP_k), (MIP_k) and $(RCSP_k)$ have different optimal values. This explains the use of the backtracking step (2) in PBS-MIP or step (4) in PBS-MIP-RCSP. The following property provides a sufficient condition to certify that no backtrack will be needed. It is a direct consequence of the fact that $(RCSP_k)$ and (MIP_k) are relaxations of (IP_k).

Property Let x_k^* denote the schedule assigned to k in an optimal solution to $RCSP(x_1^*, \cdots, x_{k-1}^*)$ or $MIP(x_1^*, \cdots, x_{k-1}^*)$. If $IP(x_1^*, \cdots, x_k^*)$ has at least one feasible solution, then there is an optimal solution to $IP(x_1^*, \cdots, x_{k-1}^*)$ that assigns schedule x_k^* to k.

Suppose the existence of a procedure, called FEASIBLE, which tries to determine a feasible solution to a given input problem (IP_k), and produces the answer "YES" if such a solution was found, and "I DON'T KNOW" otherwise.

Since ($RCSP_k$) and (MIP_k) are much easier to solve when compared to (IP_k), we propose to first determine a schedule for k by solving ($RCSP_k$), and then check, using the FEASIBLE procedure, whether the resulting (IP_k) is feasible. If we are not able to exhibit a feasible solution to (IP_k) (i.e., FEASIBLE(IP_k)="I DON'T KNOW"), then we solve the corresponding (MIP_k) which is still easier to solve than (IP_k). Again, if the resulting (IP_k) can be proved feasible, this proves that the schedule assigned to k is the best possible one; otherwise we solve (IP_k). It often happens that (IP_k) and ($RCSP_k$) have different optimal values when k is large. This is due to the fact that the number of junior employees becomes critical to cover the residual pairings. We have therefore decided to solve ($RCSP_k$) only up to employee k_{max}, where k_{max} is a fixed parameter. This process, called PBS-FeasibleIP, is summarized in Figure 5.

Procedure PBS-FeasibleIP

(0) Set k:=1;

(1) if $k \leq k_{max}$ then go to (2) else go to (3);

(2) determine an optimal solution s_{RCSP} to $RCSP(x_1^*, \cdots, x_{k-1}^*)$;
let x_k^* denote the schedule assigned to k in s_{RCSP};
if FEASIBLE($IP(x_1^*, \cdots, x_k^*)$) = "I DON'T KNOW" then go to (3) else go to (5)

(3) determine an optimal solution s_{MIP} to $MIP(x_1^*, \cdots, x_{k-1}^*)$;
let x_k^* denote the schedule assigned to k in s_{MIP};
if FEASIBLE($IP(x_1^*, \cdots, x_k^*)$) = "I DON'T KNOW" then go to (4) else go to (5)

(4) set x_k^* equal to the schedule of k in an optimal solution of $IP(x_1^*, \cdots, x_{k-1}^*)$;

(5) if $k < m$ then set $k := k + 1$ and go to (1) else STOP.

Figure 5: a new heuristic sequential algorithm for the PBS

For illustration, both ($RCSP_1$) and (MIP_1) suggest to assign pairing G to employee 1 in the example of Section 2, while this should not be done since, as already observed, this will induce future backtrack. The new proposed algorithm PBS-FeasibleIP does not assign pairing G to employee 1 because, with such a schedule x_1^*, the output of procedure FEASIBLE($IP(x_1^*)$) at Steps 2 and 3 is "I DON'T KNOW".

In the next section, we show that the output of procedure FEASIBLE can be obtained by solving a graph coloring problem. A tabu search algorithm for this problem is proposed in Section 5.

4 A graph coloring model

Given a graph $G = (V, E)$ with vertex set V and edge set E, a coloring of G is a function $c : V \rightarrow \mathbb{N}$. The value $c(x)$ of a vertex x is called the color of x. If two adjacent vertices

x and y have the same color i, the edge linking x with y is said *conflicting*. A coloring without conflicting edges is said *legal*. The Graph Coloring Problem (*GCP* for short) is to determine a legal coloring with the smallest number of different colors, called *chromatic number* of G.

The *GCP* is NP-hard. Although many exact algorithms have been designed for this problem, such algorithms can only be used to solve small instances. Heuristic algorithms, on the other hand, can be used on much larger instances, but only to produce upper bounds on the chromatic number.

In this section, we describe a graph coloring model for checking whether (IP_k) contains at least one feasible solution. To each problem (IP_k) we associate a graph G_k which is constructed as follows:

- for each pairing p, we create b_p^k vertices denoted $v_{\alpha p}$ $(\alpha = 1, \cdots, b_p^k)$; in the following we denote $V_p = \{v_{1p}, \cdots, v_{b_p^k p}\}$;
- two vertices $v_{\alpha p}$ and $v_{\beta q}$ are linked by an edge if and only if $p{=}q$ or p overlaps with q.

The graph G_1 associated with (IP_1) for the example of Figure 3 is represented in Figure 6.

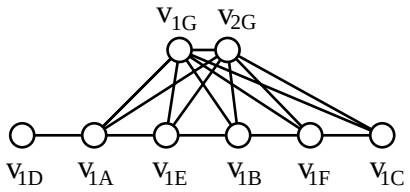

Figure 6: The graph associated with the pairings of Figure 3

We associate a color i to each employee $i = k, \cdots, m$, and a coloring of the vertices of G_k is defined as a function c that assigns exactly one color $c(v)$ to each vertex v. Such a definition means that a coloring satisfies constraints (2), (3) and (4) of (IP_k).

To simplify the presentation, we suppose here below that the sets S_i^k $(i = k, \cdots, m)$ of feasible schedules at iteration k are defined with the following constraints:

(a) a schedule cannot contain overlapping pairings
(b) all pre-assignments requirements must be fulfilled
(c) the workload (i.e., the total number of flight credits) in the schedule assigned to employee i must lie in the interval $[L_i, U_i]$.

As will be discussed in Section 7, other constraints (e.g., collective agreements between the company and the employees, airline security rules) can easily be added to the model.

A conflicting edge (i.e., an edge having both endpoints with the same color) corresponds to the situation where an employee performs two tasks $v_{\alpha p}$ and $v_{\beta p}$ of the same pairing p, or two tasks $v_{\alpha p}$ and $v_{\beta q}$ in two overlapping pairings p and q. Hence, a legal coloring of G_k satisfies constraints (a).

We assign a list $L(v)$ of possible colors to each vertex v. If employee i has to be pre-assigned to pairing p, then we choose one vertex $v_{\alpha p} \in V_p$ and we restrict the set $L(v_{\alpha p})$ of possible colors for this vertex to the singleton $\{i\}$. Any task v that is not pre-assigned to an employee i has $L(v) = \{k, \cdots, m\}$.

Constraints (b) are taken into account by imposing that a coloring c must assign a color $c(v) \in L(v)$ to each vertex v. In our example, we impose that v_{1D} must have color 3 (i.e., $L(v_{1D}) = \{3\}$).

Let w_p denote the number of flight credits of pairing p. We assign a weight denoted $w(v)$ and equal to w_p to each vertex $v \in V_p$. Constraints (c) impose that the total weight $\sum_{c(v)=i} w(v)$ of vertices with color i is a number in $[L_i, U_i]$ for all $i = k, \cdots, m$.

The schedules are linked together with constraint (1) of (IP_k) imposing that all qualification requirements must be fulfilled. These constraints can be interpreted in terms of graph coloring. Indeed, let I_q denote the set of employees in $\{k, \cdots, m\}$ having qualification q. To satisfy constraint (1), a coloring must be such that at least N^k_{pq} vertices in V_p must have a color in I_q. In our example, we impose that at least one vertex among v_{1G} and v_{2G} must have color 1 or 3.

In summary, our goal is to determine a legal coloring c of G_k such that

- $c(v) \in L(v)$ for all vertices v,
- $\sum_{c(v)=i} w(v) \in [L_i, U_i]$ $(i = k, \cdots, m)$,
- at least N^k_{pq} vertices in V_p have a color in I_q.

Such a coloring exists if and only if (IP_k) has a feasible solution.

Finding a legal coloring G_k (i.e., a coloring that satisfies constraints (a)) or proving that no such coloring exists is an easy task. Indeed, the graph G_k is an interval graph (Hajós, 1957), and polynomial algorithms are known to color such graphs. However the above additional requirement make the problem much more difficult. Indeed, for example, it is shown in (Biró et al. 1992) that two pre-assignments are sufficient to transform the coloring problem from polynomially solvable to NP-hard. We therefore propose a heuristic coloring algorithm which is presented in the next section.

5 A tabu search algorithm

Let S be the set of solutions to a combinatorial optimization problem. For a solution $s \in S$, let $N(s)$ denote the *neighborhood* of s which is defined as the set of *neighbor solutions* in S

obtained from s by performing a *local change* on it. Local search techniques visit a sequence $s_0, \ldots, s_n$ of solutions, where s_0 is an initial solution and $s_{i+1} \in N(s_i)$ $(i = 1, \ldots, n-1)$. *Tabu search* is one of the most famous local search technique. It was introduced by Glover in 1986. A description of the method and its concepts can be found in Glover and Laguna (1997). A basic tabu search is described in Figure 7.

Tabu Search

Choose an initial solution s; set $TL = \emptyset$ (tabu list); set $s^* = s$ (best solution);

Repeat the following until a stopping criterion is met

- Determine a best solution $s' \in N(s)$ such that either $s' \notin TL$ or s' is better than s^*;
- If s' is better than s^* then set $s^* := s'$;
- Set $s := s'$ and update TL;

Figure 7: Basic tabu search

We describe in this section a tabu search algorithm for our graph coloring problem. Since we follow the general scheme of Figure 7, it is sufficient to define the solution space S, the objective function f, the neighborhood N and the tabu list TL.

The proposed tabu search algorithm is an extension of the *Tabucol* algorithm described in (Hertz et al., 1987). *Tabucol* first generates an initial coloring which contains typically a large number of conflicting edges. Then, the heuristic iteratively modifies the color of a single vertex, the objective being to decrease progressively the number of conflicting edges until a legal coloring is obtained.

Our adaptation works as follows. We define the solution space S as the set of colorings c such that $c(v) \in L(v)$ for all v in G_k and such that $c(v_{\alpha p}) \neq c(v_{\beta p})$ for all $p \in P$ and $\alpha \neq \beta$. This means that our algorithm will only visit colorings that satisfy the pre-assignment requirements and such that each employee performs at most one task in each pairing p.

The objective function we try to minimize is a weighted sum of three components. More precisely, given a coloring c, we define $f(c) = \omega_1 f_1(c) + \omega_2 f_2(c) + \omega_3 f_3(c)$, where ω_1, ω_2 and ω_3 are penalty parameters that give more or less importance to each component of $f(c)$, and $f_1(c), f_2(c)$ and $f_3(c)$ are defined as follows.

- $f_1(c)$ is the number of conflicting edges. This value corresponds to the number of overlapping situations.
- Let $W_i(c)$ denote the total weight $\sum_{c(v)=i} w(v)$ of the vertices with color i $(i = k, \cdots, m)$. We define $f_2(c) = \sum_{i=k}^{m} max\{0, L_i - W_i(c), W_i(c) - U_i\}$. This value is a total penalty for the situations where $W_i(c) \notin [L_i, U_i]$.

- For a paring $p \in P$ and a qualification $q \in Q$, let $n_{pq}(c)$ denote the number of vertices $v \in V_p$ with $c(v) \in I_q$. We define $f_3(c) = \sum_{p \in P} \sum_{q \in Q} max\{0, N^k_{pq} - n_{pq}(c)\}$. This value corresponds to the number of violations of qualification requirements.

Parameters ω_i are initially set equal to 1 and are then adjusted every 10 iterations, as in (Gendreau et al., 1994): if the ten previous solutions c had a value $f_i(c) = 0$ then ω_i is divided by 2; if these 10 values were all strictly positive, then ω_i is multiplied by 2; otherwise ω_i remains unchanged.

A move from a solution to a neighbor one consists in changing the color of a single vertex. For a vertex v and a color $i \neq c(v)$, we denote (v, i) the move that assigns color i to v, and we denote $c + (v, i)$ the solution resulting from this move. Let $p(v)$ denote the pairing associated with vertex v (i.e., $v \in V_{p(v)}$). Since an employee cannot perform more than one task in each pairing, we do not consider moves (v, i) such that there is a vertex $u \neq v$ with $p(v) = p(u)$ and $c(u) = i$. Also, in order to satisfy the pre-assignment requirements, we impose that $i \in L(v)$ for all moves (v, i). In summary, the neighborhood $N(c)$ of a solution $c \in S$ is defined as the set of solutions c' that can be reached from c by applying a move (v, i) with $i \in L(v)$ and $i \neq c(u)$ for all $u \in V_{p(v)}$.

The tabu list TL stores forbidden moves. More precisely, when a move (v, i) is performed on a solution c, the tabu list stores the pair $(v, c(v))$, which means that it is forbidden to reassign color $c(v)$ to v for a given number of iterations.

There are typically many vertices v than can receive a new color i without inducing any change in the objective function (i.e., $f(c) = f(c + (v, i))$. In order to guide the search towards good regions of the search space S, we try to force the algorithm to deal with those vertices that are responsible for the current strictly positive value of $f(c)$. This is done by considering only *interesting* moves which are defined as follows.

Definition Let TL be a tabu list and c^* the current best solution. A move (v, i) from a solution c is *interesting* if $(v, i) \notin TL$ or $f(c) < f(c^*)$, and at least one of the following properties is satisfied:

- vertex v is one of the endpoints of a conflicting edge (and is therefore responsable for the strictly positive value of $f_1(c)$).
- $W_{c(v)}(c) > U_{c(v)}$. In such a case, employee $c(v)$ has too many flight credits and we reduce $f_2(c)$ by removing one task from this employee.
- $W_i(c) < L_i$. In such a case, employee i does not have enough flight credits and we reduce $f_2(c)$ by adding one task to this employee.
- there is a pairing $p \in P$ and a qualification $q \in Q$ such that $v \in V_p$, $\delta_{c(v)q} = 0$ (i.e., $c(v)$ does not have qualification q), and $N^k_{pq} > n_{pq}(c)$ (i.e., the number of employees assigned to p with qualification q is not large enough). In such a case $f_3(c) > 0$ and the algorithm tries to assign a new employee i to v with qualification q (i.e., $i \in I_q$).

In order to build an initial solution, we follow two strategies. If $k = 1$ then we build an initial coloring of G_k by using a greedy algorithm that simply colors the vertices one by one, trying to avoid creating conflicting edges as well as exceeds in the workloads. An initial coloring in $G_k(k > 1)$ is simpler to obtain since a coloring c_{k-1} of G_{k-1} is known. Indeed, G_k is a subgraph of G_{k-1} obtained by removing those vertices corresponding to the tasks assigned to employee $k-1$. An initial coloring c_k of G_k can simply be obtained by setting $c_k(v) = c_{k-1}(v)$ if $c_{k-1}(v) \neq k-1$, and by using the above greedy algorithm only for those vertices v in G_k with $c_{k-1}(v) = k-1$. A detailed description of this procedure is given in Figure 8.

Procedure INIT
if $k = 1$ then
  set $W_i = 0$ for all $i = 1, \cdots, m$, and $c(v) = 0$ for all v in G_1
else
  let $c_{k-1}(v)$ denote the color of v in G_{k-1};
  for all vertices v do
    if $c_{k-1}(v) = k-1$ then set $c(v) = 0$
    else set $c(v) = c_{k-1}(v)$;
for all vertices v with $c(v) = 0$ do
  for all i in $L(v)$ do
    If $c(u) = i$ for some $u \in V_{p(v)}$ then set $value(i) = \infty$
    else
      let μ_{iv} denote the number of vertices u adjacent to v with color $c(u) = i$;
      set $value(i) = \mu_{iv} + max\{0, W_i + w(v) - U_i\}$
  Choose a color i for v with minimum $value(i)$ and add $w(v)$ units to W_i

Figure 8: Procedure that generates an initial coloring in G_k

Our tabu search algorithm is described in Figure 9. At each iteration we perform the best interesting move (ties are broken randomly). The algorithm stops as soon as $f(c) = 0$, or after a given time limit.

Algorithm Tabu-PBS-FIP
build an initial solution s according to procedure INIT in Figure 8;
set $c^* := c$, $iter = 0$ and $TL = \emptyset$;
repeat until $f(c) = 0$ or a given time limit is reached

- choose an interesting move (v, i) with minimum value $f(c + (v, i))$;
- if $f(c) < f(c^*)$ then set $c := c + (v, i)$, and update TL;
- set $c^* := c$;

Figure 9: Our tabu search algorithm

Notice that the value $f_1(c+(v,i))$ and $f_2(c+(v,i))$ of a move (v,i) from c can be computed in constant time, while $f_3(c+(v,i))$ can be computed in $O(\mid Q \mid)$ time. Such a complexity can be achieved by storing the following information:

- the number μ_{ju} of vertices with color j adjacent to u, for all j and u,
- the workload W_j of each employee i,
- the number $n_{pq}(c)$ of vertices $u \in V_p$ with $c(u) \in I_q$, for all p and q.

Indeed,

- $f_1(c+(v,i)) = f_1(c) + \mu_{iv} - \mu_{c(v)v}$,
- $f_2(c+(v,i)) = f_2(c) - max\{0, L_i - W_i(c), W_i(c) - U_i\}$
  $-max\{0, L_{c(v)} - W_{c(v)}(c), W_{c(v)}(c) - U_{c(v)}\}$
  $+max\{0, L_i - W_i(c) - w(v), W_i(c) + w(v) - U_i\}$
  $+max\{0, L_{c(v)} - W_{c(v)}(c) - w(v), W_{c(v)}(c) + w(v) - U_{c(v)}\}$
- $f_3(c+(v,i)) = f_3(c) - \sum_{q\in Q} max\{0, N^k_{pq} - n_{pq}(c)\}$
  $+ \sum_{q\in Q} max\{0, N^k_{pq} - n_{pq}(c) - \delta_{iq} + \delta_{c(v)q}\}$

Once a move (v,i) is performed, it is sufficient to do the following update:

- for all vertices u adjacent to v do: add 1 to μ_{iu} and remove 1 from $\mu_{c(v)u}$
- add $w(v)$ units to W_i and remove $w(v)$ units from $W_{c(v)}$
- for all $q \in Q$ do: add 1 unit to $n_{p(v)q}(c)$ if $i \in I_q$ while $c(v) \notin I_q$, and remove 1 unit from $n_{p(v)q}(c)$ if $c(v) \in I_q$ while $i \notin I_q$.

6 Model validation on a real instance

To validate the proposed model and algorithm, we have performed experiments on a real instance from a North American carrier. This instance has been chosen for two reasons. First, the constraints that have to be taken into account in this instance are exactly those considered in this paper, while other airline companies have additional constraints which have not yet been included in our model. Second, this instance is known for being difficult to solve and requiring several backtracks when using the PBS-MIP-RCSP algorithm described in Figure 4. The instance contains 330 employees, 1956 pairings, and $\mid Q \mid = 11$ different qualifications. The lower and upper bounds L_i and U_i on the workload of each employees are 65 and 90 hours, respectively.

When using the PBS-MIP-RCSP algorithm of Figure 4, a first backtrack occurs at employee 248. The wrong decision is taken for employee 218, which means that the backtracking process (i.e., step (4) in PBS-MIP-RCSP) demonstrates the infeasibility of $(IP_{247}), (IP_{246}), \cdots, (IP_{219})$ and then finds an optimal solution to (IP_{218}). A second back-

track occurs for employee 303 and it is then discovered that a wrong decision was taken for employee 223.

We have tested our PBS-FeasibleIP strategy with a time limit of 30 seconds for each tabu search. This means that if we are not able to produce a coloring c with $f(c) = 0$ in less than 30 seconds, we conclude that the current (IP_k) probably has no feasible solution. We have decided to solve $(RCSP_k)$ only up to employee $k_{max} = 200$. For an employee $k > k_{max}$, we skip step (2) and directly solve (MIP_k) and test if the resulting (IP_{k+1}) is feasible.

When solving our real instance, we have run each tabu search 50 times, and all runs have given the same results. The employees can be divided into four groups:

(a) For $k \leq 200$, $k \neq 165, 195$, we have determined a schedule by solving $(RCSP_k)$, and we have got the output "YES" from FEASIBLE(IP_{k+1}).

(b) For $k = 165, 195$, we have first determined a schedule by solving $(RCSP_k)$; since the tabu search algorithm was not able to color the resulting G_{k+1} (i.e., we got the answer "I DON'T KNOW" from FEASIBLE(IP_{k+1})), we have then solved (MIP_k); we were finally able to color the resulting G_{k+1} (i.e., we got the output "YES" from FEASIBLE(IP_{k+1})).

(c) For $k > 200$, $k \neq 218, 223$, we have determined a schedule by solving (MIP_k) and we have got the output "YES" from FEASIBLE(IP_{k+1}).

(d) For $k = 218, 223$, we have first determined a schedule by solving (MIP_k); since the tabu search algorithm was not able to color the resulting G_{k+1} (i.e., we got the answer "I DON'T KNOW" from FEASIBLE(IP_{k+1})), we have then solved (IP_k).

Our tabu search needs more time to color G_1 than a G_k with $k > 1$. The reason is that when coloring G_1, the INIT procedure of Figure 8 must build an initial solution starting from scratch, and this initial solution typically contains many violations of constraints. A coloring of $G_k (k > 1)$ is much easier to obtain since the initial solution is obtained by copying the coloring c_{k-1} of G_{k-1}, and then assigning a new color $> k - 1$ to the vertices which had color $k - 1$ in c_{k-1}. In our experiments, we often got initial feasible colorings c with $f(c) = 0$, which means that no iteration of the tabu search was needed.

We have colored graph G_1 of our instance in an average of 22 seconds (the average being taken over the 50 runs). The graphs G_k with $k > 1$ for which we have been able to determine a coloring c with $f(c) = 0$ have been colored, in average, in less than one second, the worst case being 4.7 seconds. In summary, our tabu search was run, in average for less than 9 minutes. Indeed, we needed 22 seconds to color G_1, exactly 2 minutes (4 times 30 seconds for employees 164, 195, 218, 223 of the above cases (b) and (d)), and less than 6 minutes to color the 325 other G_k's. The time limit of 30 seconds seems reasonable since all colorings c with $f(c) = 0$ were obtained in less than 5 seconds (except for G_1). All

these times can be considered as negligible when compared to the hours needed to solve the (IP_k)'s.

Notice finally that we have solved only 2 (IP_k)'s (one for employee 218, and one for employee 223), while the PBS-MIP-RCSP algorithm required the solution of 110 (IP_k)'s: $(IP_{247}), \cdots, (IP_{218})$ during the first backtrack and $(IP_{302}), \cdots, (IP_{223})$ during the second backtrack. This is a huge saving of time.

7 Final remarks

We have proposed a graph coloring model and a tabu search algorithm for solving a feasibility problem in crew scheduling. Our algorithm either produces a feasible schedule for each employee while covering all pairings, or terminates with the message "I DON'T KNOW", which means that no proof of feasibility or infeasibility could be produced. Our algorithm can be efficiently combined with the heuristic sequential algorithm for the PBS proposed by Gamache et al. (1998). This algorithm takes decisions which may lead to backtracks, and these backtracks can be avoided with the help of our solution method.

We have tested our method on two instances that were known for being difficult to solve because of the occurrences of several backtracks in the solution process. We have shown with these two examples that the cost of avoiding these backtracks is a few minutes to be compared to hours that can be saved. In fact, our algorithm only needs a few seconds to prove that no backtrack will be needed. More time is only needed when a future backtrack is suspected, in which case we don't take any chance and therefore solve the current (IP_k).

The main objective of our research project was to validate the use of a graph coloring model and a tabu search algorithm for solving a feasibility problem in crew scheduling. As shown in this paper, the proposed model can easily handle qualifications and workload constraints. This list of constraints represents the ones that are most commonly encountered in the airline industry, but should however not be considered as exhaustive. Several other constraints can be added to the model. Among them, we mention the constraints related to the assignment of rest periods to employees, airline security rules, and collective agreements between the company and the employees. Since such constraints differ significantly from one company to the other, they have not been considered in the first phase of this research project. We are however confident that these particular constraints can be integrated into our model.

Our method can be considered as flexible since it offers the possibility to handle each requirement either as a hard or a soft constraint. For example, all constraints imposed on the schedule of an employee correspond to constraints on a colors class (i.e., a set of vertices with the same color). One can decide to consider some of these constraints as hard, which means that all colorings visited by our tabu search satisfy theses constraints: this is what we did with the pre-assignment requirements which are never violated. One

can prefer to accept colorings that violate some of these constraints: in such a case, one must penalize the constraint violations, as was done with function f_2 for the violations of the maximal and minimal workloads.

Future works will concentrate on the integration in our model of particular constraints that have been omitted in this initial phase of our research project (e.g. constraints related to a limited set of patterns for the rest periods, and constraints on the sequence of workloads between rest periods).

References

[1] Achour, H., M. Gamache and F. Soumis Branch and Cut at the Subproblem Level in a Column Generation Approach: Application to the Airline Industry. *Cahier du GERAD G-2003-34*.

[2] Biró, M., M. Hujter and Zs. Tuza (1992). Precoloring extension I: interval graphs. *Discrete Mathematics* 100(1-3), 267–279.

[3] Byrne, J. (1988). A Preferential Bidding System for Technical Aircrew. *1988 AGIFORS Symposium Proceedings* 28, 87–99.

[4] El Idrissi, T. (2002). *Amélioration de la méthode des compteurs pour la construction des blocs mensuels personnalisés d'agents de bord*. Master Thesis École Polytechnique, Montréal.

[5] Federici, F. and D. Paschina (1990). Automated Rostering Model. *AGIFORS Symposium Proceedings* 26.

[6] Gamache, M. and F. Soumis (1998a). A Method for Optimally Solving the Rostering Problem. In: *Operations Research in the Airline Industry*, G. Yu (ed.), Kluwer, Norwell, MA, 124–157.

[7] Gamache, M., F. Soumis, J. Desrosiers, D. Villeneuve, and E. Gélinas (1998b). The Preferential Bidding System at Air Canada. *Transportation Science* 32, 246–255.

[8] Gamache, M., F. Soumis, G. Marquis, and J. Desrosiers (1999). A Column Generation Approach for Large-Scale Aircrew Rostering Problems. *Operations Research* 47, 247–262.

[9] Gendreau, M., A. Hertz and G. Laporte (1994), A tabu search heuristic for the vehicle routing problem. *Management Science* 40, 1276-1290.

[10] Giafferri, C., J.P. Hamon, and J.G. Lengline (1982). Automatic Monthly Assignment of Medium-Haul Cabin Crew. *1982 AGIFORS Symposium Proceedings* 22, 69–95.

[11] Glanert, W. (1984). A Timetable Approach to the Assignment of Pilots to Rotations. *1984 AGIFORS Symposium Proceedings* 24, 369–391.

[12] Gontier, T. (1985). Longhaul Cabin Crew Assignment. *1985 AGIFORS Symposium Proceedings* 25, 44–66.

[13] Hajós G. (1957). Über eine Art von Graphen. *INt. Math. Nachrichtern* 11.

[14] Hertz A. and D. de Werra (1987) Using Tabu Search Techniques for Graph Coloring. *Computing* 39, 345-351, 1987.

[15] Jeandroz, P.(1999) *Heuristique pour la construction de blocs mensuels personnalisés d'agentsde bord*. Master Thesis École Polytechnique, Montréal.

[16] Marchettini, F. (1980). Automatic Monthly Cabin Crew Rostering Procedure. *1980 AGIFORS Symposium Proceedings* 20, 23–59.

[17] Mayer, M. (1980). Monthly Computerized Crew Assignment. *1980 AGIFORS Symposium Proceedings* 20, 93–124.

[18] Moore, R., J. Evans, and H. Ngo (1978). Computerized Tailored Blocking. *1978 AGIFORS Symposium Proceedings* 18, 343–361.

[19] Nicoletti, B.(1975). Automatic Crew Rostering. *Transportation Science* 9, 33–42.

[20] Ryan, D.M. (1992). The Solution of Massive Generalized Set Partitioning Problems in Air Crew Rostering. *Operations Research* 45, 649–661.

[21] Schmidt, W.R. and J. Hosseini (1994). Preferential Schedule Assignments for Airline Crew Scheduling. *ORSA/TIMS conference*, Detroit.

[22] Sherali, H.D. and A.L. Soyster (1983). Preemptive and Non preemptive Multi-Objective Programming: Relationships and Counterexamples. *Journal Of Optimisation Theory and Applications* 39, 173–186.

[23] Tingley, G.A. (1979). Still Another Solution Method for the Monthley Aircrew Assignment Problem. *1979 AGIFORS Symposium Proceedings* 19, 143–203.