European Journal of Operational Research 199 (2009) 674–683

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Genetic algorithm based approach for the integrated airline crew-pairing and rostering problem

Nadia Souai *, Jacques Teghem

Laboratory of Mathematics and Operations Research, Faculty of Engineering, Mons, 09, Rue de Houdain, 7000 Mons, Belgium

ARTICLE INFO

Article history:
Received 22 September 2006
Accepted 8 October 2007
Available online 13 April 2008

Keywords:
Airline application
Combinatorial optimization
Genetic algorithm
Heuristics

ABSTRACT

Airline crew scheduling problem is a complex and difficult problem faced by all airline companies. To tackle this problem, it was often decomposed into two subproblems solved successively. First, the airline crew-pairing problem, which consists on finding a set of trips – called pairings – i.e. sequences of flights, starting and ending at a crew base, that cover all the flights planned for a given period of time. Secondly, the airline crew rostering problem, which consists on assigning the pairings found by solving the first subproblem, to the named airline crew members. For both problems, several rules and regulations must be respected and costs minimized.

It is sure that this decomposition provides a convenient tool to handle the numerous and complex restrictions, but it lacks, however, of a global treatment of the problem. For this purpose, in this study we took the challenge of proposing a new way to solve both subproblems simultaneously. The proposed approach is based on a hybrid genetic algorithm. In fact, three heuristics are developed here to tackle the restriction rules within the GA's process.

1. Introduction

Airline crew scheduling problem is a complex and difficult problem faced by all airline companies. Indeed, the cost related to the crew members constitutes the most important direct cost supported by the airline company, after the fuel cost.

To tackle this problem, it was often decomposed into two subproblems solved successively. First, the airline crew-pairing problem, which consists on finding a set of trips – called pairings – i.e. sequences of flights, starting and ending at a crew base, that cover all the flights planned for a given period of time. Secondly, the airline crew rostering problem, which consists on assigning the pairings found by solving the first subproblem, to the named airline crew members. For both problems, several rules and regulations must be respected and costs minimized.

In our recent Ph.D. thesis [9], we first tackled successively these two subproblems by means of a hybrid genetic algorithm.

The formulation of the first problem – the airline crew pairing – was based on its decomposition according to the set of duty periods instead of pairing. We recall that a duty period is an one day duration sequence of flights, i.e. an one day pairing; then the problem was solved by an extension and an improvement of Levine's algorithm [3]. For the second problem – the airline crew rostering – we proposed a decomposition model to reduce the model complexity related to its large size even for small size instances.

Indeed, a new idea was applied, where only legal weekly rosters are generated and assigned and the required monthly restrictions are checked simultaneously. The resolution method used is a genetic algorithm combined, this time, with a Simulated Annealing approach. Even experimented on few instances, this innovating approach gives interesting results [10].

The following Fig. 1 illustrates the decomposed airline crew scheduling process.

The second part of the thesis [9] is devoted to the simultaneous treatment of the two problems as described in the present paper. The motivation is double.

On the one hand, from a practical point of view, dealing with these two dependant problems separately, lacks of a good estimation of the real cost of the global schedule. Indeed, in the airline crew-pairing problem, the total duration of the set of selected pairings is minimized; however the real cost of a planning must be calculated after the assignment of the pairings to the different crew members is accomplished – i.e. after the airline crew rostering problem is solved – since the crew members are payed a fixed salary for a given working time and an additional cost for each exceeding hour.

On the other hand, the "optimal" set of pairings, provided by the resolution of the airline crew-pairing problem, is determined without considering the number and the availabilities of the different crew members which are, also, known afterward while dealing with the airline crew rostering problem.

To avoid these drawbacks, in this paper we attempt to propose an effective approach to integrate the airline crew-pairing and

* Corresponding author. Tel.: +32 65 37 46 84; fax: +32 65 37 46 89.
E-mail address: nadia.souai@fpms.ac.be (N. Souai).

0377-2217/$ - see front matter © 2008 Published by Elsevier B.V.
doi:10.1016/j.ejor.2007.10.065

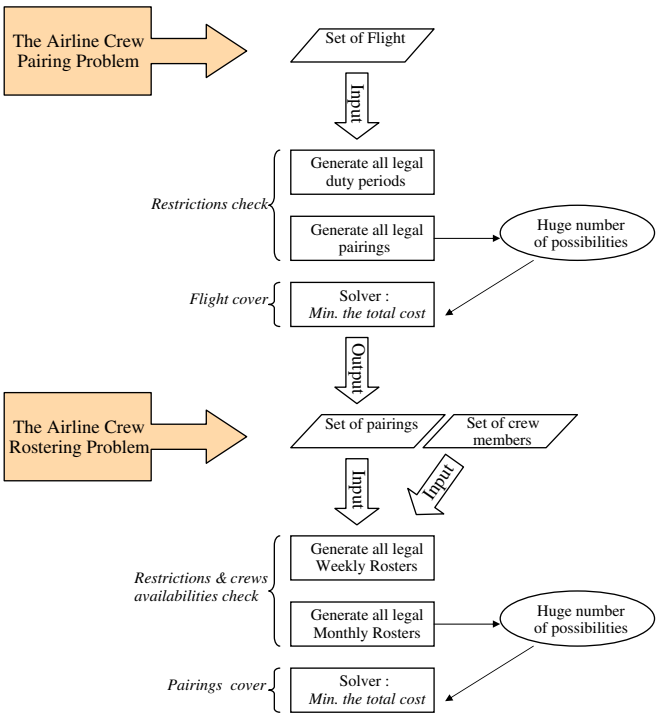

Fig. 1. Airline crew scheduling.

rostering problems in a single model and thus to solve them simultaneously.

Again we proposed a genetic algorithm but clearly this one is very different of those used to solve the two problems separately, due to a completely different representation of the chromosomes and thus a different initialization process and different operators. For the same reason, the local search heuristics to which the genetic algorithm is hybridized are news.

This paper is organized as follows: in Section 2 a brief overview of the problem and the literature is presented. In Section 3 the different steps of the approach, combining a genetic algorithm with different local search heuristics, are summarized. Section 4 is devoted to the detailed description of the applied genetic algorithm, followed by the description of the local search heuristics in Section 5. In Section 6 some computational results will be presented.

2. Description of the problem

Basically, the airline crew scheduling problem is defined as the problem of assigning a set of flights of a given kind of aircraft (the aircraft routing problem), to a set of crew members of the same category (in our case technical crew members: pilot, first officer and instructor) able to fly this aircraft. Each crew member has a personalized calendar of availability which takes into consideration a set of previously assigned tasks, such like, training periods, vacations, medical visits, annual leaves, etc.

2.1. The restriction rules

Several restrictions, in relation with the security rules, have to be checked, namely:

- *Daily restrictions*: they are security rules that have to be respected to build legal duty periods (i.e. one day duration sequence of flights):
  - the city of arrival of a flight is the same city of departure of the flight that succeed it in the same duty period.
  - the *sit* time between the consecutive flights must be within a prescribed minimum and maximum sit times called *minsit* and *maxsit*, respectively.
  - the *elapsed* time of a duty period, including the *briefing*[1] and *debriefing* times, must be less than a maximum allowed value called *maxelapse* which varies with the departure time of the first flight in the duty period called *earlytime*.
  - the total flying duration a duty period, called *fly* cannot exceed a maximum allowed flying time called *maxfly*.
  - the duty period must be composed of up to a maximum number of flight segments called *maxlegs*.
- *Pairings restrictions*: pairings are sequences of duty periods going from one to three days of duration such that:
  - the city of arrival of a duty period is the same city of departure of the duty period that succeed it in the same pairing.
  - each pairing must begin and end at the same crew base[2] (“*Base-to-Base pairings*”). However, if this condition cannot be verified, the pairing will contain “deadhead” flights whose role is to take the crews to their respective bases at the end of a pairing, or from their bases in the beginning of the pairing.
  - the number of duty periods must be less than a maximum number of duty periods called *maxduties*.
  - the rest time between the consecutive duty periods of a pairing must be within prescribed minimum and maximum rest times called *minrest* and *maxrest*, respectively.
  - the *elapsed* time of a pairing called *TAFB* (time away from base) cannot exceed a maximum number of days called *maxday*.
- *Weekly restrictions*: all restriction rules required for the construction of legal personalized planning, of one week duration, for each crew members. A weekly roster is a sequence of pairings such that:
  - the pairings and tasks within the same rosters must not overlap.
  - the total flying time of all pairings that make up the roster must not exceed a maximum allowed flying time per week called $T_{\max,w}$.
  - each weekly roster must contain at least one free day (*LWR*: Legal Weekly Rest).
- *Monthly restrictions*: all the restrictions required to build a legal monthly personalized schedule for each crew member. Thus, a monthly roster is a sequence of weekly rosters such that:
  - the weekly rosters within the same schedule must not overlap.
  - the total flying time of all pairings that make up the schedule must not exceed a maximum allowed flying time per month called $T_{\max,m}$.
  - the crew members receive a fixed wage for a minimum guaranteed number of working hours per month, called T_g, and a supplementary amount for each exceeding flying hour.

Dealing with the integrated airline crew-pairing and rostering problem means taking into consideration all the outlined restrictions simultaneously.

[1] The *Brief* and *Debrief* times, which are fixed by the airline company, are the duration times necessary to a crew member to, respectively, begin and end his/her working day.

[2] Bases are the cities (airports) where the crew members are stationed.

2.2. The cost structure

By definition, the cost CD_l, expressed in time, of a duty period l is defined by

$$CD_l = \max\{mg1, f1 * elapsed_l, fly_l\},$$

where

- "$mg1$" is the minimum guaranteed number of hours;
- "$f1$" is a fraction of the elapsed time ("$elapsed_l$") of the duty period;
- "fly_l" is the flying time in the duty period.

On the other hand, the cost CP_p of a pairing p, also expressed in time, is the maximum of three quantities:

- a minimum guarantee "$mg2$" times the number of duties in the pairing NDP_p;
- a fraction "$f2$" of the total elapsed time of the pairing "$TAFB_p$";
- the sum of the costs of the duties that make up the pairing.

$$CP_p = \max\left\{NDP_p * mg2, f2 * TAFB_p, \sum_{d\in p} CD_d\right\}.$$

Usually, the following values are taken: $mg1 = 3$ hours, $f1 = 4/7$, $mg2 = 3$ hours and $f2 = 2/7$.

Hence, let X be a solution of the problem, i.e. a set of legal schedules for each crew member. We note by $PR_k(X)$ the set of pairings assigned to the crew member k within the solution X. The cost $cost_k(X)$ related to the crew member k is defined by the following formulation:

$$cost_k(X) = \max\left\{0, \left(\sum_{p\in PR_k(X)} CP_p\right) - T_g\right\} \times HS,$$

where HS is the monetary amount that the airline company must pay for each exceeding flying hour. The total cost of a solution X is thus

$$cost(X) = \sum_{k\in K} cost_k(X),$$

where K is the set of all considered crew members.

2.3. The deviation function of a solution

It is always desirable that the total flying time (the workload) is well distributed, among the different crew members, in an equitable way.

If we denote by

- $RF_k(X)$ the real flying time of the crew member k according to the solution X.
- AVf the average, or ideal, flying time for any crew member, i.e.

$$AVf = \frac{\sum_{k\in K} RF_k(X)}{|K|},$$

we define the average deviation of a solution X as the function:

$$DV(X) = \sum_{k\in K} |RF_k(X) - AVf|.$$

2.4. The literature

The reader can refer to [9,10], or to the recent survey [1], for the literature related to the separated treatment of each of the two problems.

Only few studies attempted to approach the integrated airline crew scheduling problem. The paper [4] presents a new idea in which the problem is formulated as a large-size mixed integer linear program based on the set of all duty periods instead of few ones. Different kind of constraints are considered, in relation with the rosters building as well as the crew members availabilities and compatibilities. However, the tackled problem is a special case of an airline company where the pairings are not built but only duty periods are considered.

In [7] a set of duty periods, that cover all the flights once, is built from the set of aircraft routes. Afterward, a genetic algorithm is applied to assign the duty periods to the different crew members. Here, a matrix-based representation of the chromosomes is used and the fitness function is the aggregation of diverse penalties in relation with the different restrictions violation. However, the main drawback of this approach lies in the fact that, beside the various penalties that make a feasible solution hard to obtain, the first set of duty periods stays unchanged during the GA iterations.

As far as our study is concerned, we apply a hybrid genetic algorithm to solve the problem. However, and unlike [7], the set of initial duty periods change during the GA process to enable the change within the structure of the duty periods themselves. In addition, a unique penalty is considered (uncovered and over-covered flights) and a local search heuristic is developed to handle it. We also introduced a second objective the average deviation.

Let us also mention two very recent studies:

- Guo et al. [2] analyzed almost the same problem but the integration is partial because their approach solved successively two coupled components: firth the building of the pairings and then the crew schedules.
- Mercier and Soumis [6] proposed an integration of two other components of the general airline schedule problem, i.e. the flight timing problem and the pairing problem.

3. Principle of the resolution scheme

3.1. Notations

The following notations will be considered throughout the paper:

J	set of the month days (index $j \in J$)
K	set of all crew members (index $k \in K$)
I	set of flight segments to be assigned (index $i \in I$): $I = \bigcup_{j\in J} I_j$, where I_j is the set of the flights of the day j
$DP(j)$	set of all legal duty periods built during the day j
I_{jl}	set of flights of the day j that constitute the duty period l

A solution X to the problem is a set of personalized schedules for each available crew member. One can represent X by the assignment of a set of feasible duty periods (i.e. a subset of $\bigcup_{j\in J} DP(j)$) to the available crew members K.

We introduce the following terminology:

3.1.1. Legality of a solution

A solution X is said "legal" if all the restrictions associated to the pairings, rosters and crew availabilities are satisfied.

In the case of illegal solution X, $L(X)$ will be the list of couples (crew, day) containing illegal assignment (i.e. a duty period which is not compatible with the duty periods already assigned to the crew member during the days $j \in J$ $s.t.$ $j \neq day$). For a legal solution X, we have $L(X) = \emptyset$.

3.1.2. Feasibility of a solution

Each flight must be covered exactly once. In case of unfeasible solution, we denote by $Pena_j(X) = ncf_j(X) + ocf_j(X)$ the penalty of the day j associated with the solution X, where

- $ncf_j(X)$ is the set of non covered flights during the day j, according to the solution X.
- $ocf_j(X)$ is the set of over-covered flights during the day j, according to the solution X.

For a feasible solution X we have $Pena(X) = \sum_{j \in J} Pena_j(X) = 0$. Obviously, the aim is to find a legal and feasible solution.

3.2. The objectives

We tackle a bi-objective optimization problem in a hierarchical way:

– the main objective is to minimize the total cost $cost(X)$;
– to differentiate between the equivalent solutions, we consider as a second objective to minimize the deviation $DV(X)$.

3.3. Structure of the algorithm (see Fig. 2)

After the generation of all possible legal duty periods, roughly speaking, the main idea of the proposed approach, is to assign some of them to the crew members according to their own availability.

After the building of the initial population of solutions in the proposed GA, at each iteration of the GA we can distinguish two main steps:

- The first step is based on a "multi-points crossover" or a "mutation" operators applied to two selected solutions.
  These operators consist mainly on reassigning certain duty periods among the different crew members while taking into account to not violate any restriction and only constraints related to the flights coverage can be breached. In other words, at this step, only non feasible solutions can possibly be considered and illegal ones could not.
  For this purpose, the GA operators are first accomplished "carefully" by checking all the required restrictions and then are combined with a local search heuristic called "Legality Repair Heuristic" whose aim is to reestablish the legality of the different pairings, and hence rosters, in order to obtain $L(X) = \emptyset$ for each considered solution X.
- In a second step, two heuristics are randomly applied to the obtained individuals in order to reduce their penalty $Pena(X)$, this time related to the flights cover, i.e. the feasibility of each solution is improved.

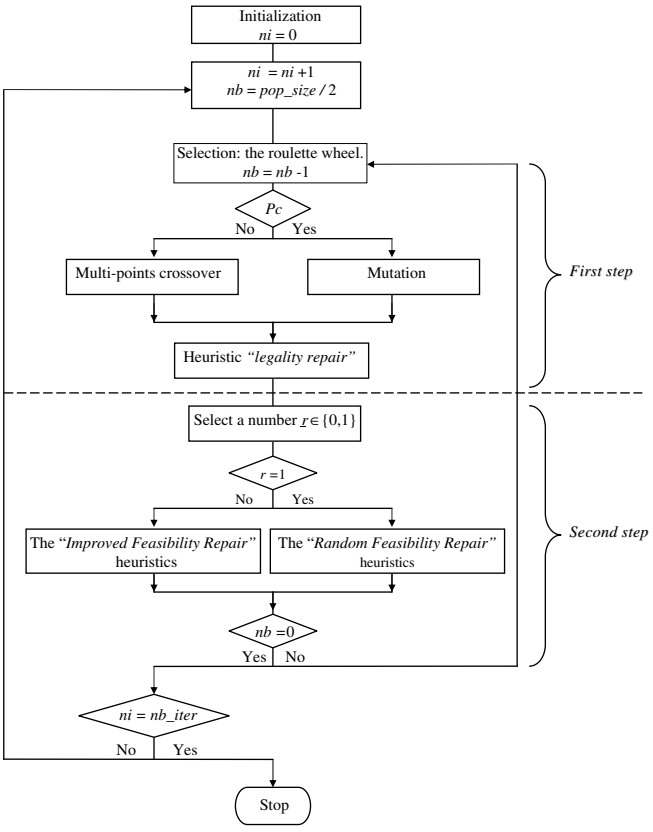

Fig. 2. The hybrid genetic algorithm process.

To consider simultaneously:

– the necessity to find feasible solutions;
– the hierarchical optimization described above in 3.2.

The proposed fitness function will contain three terms, also considered hierarchically in the following order:

- first the penalty $Pena(X)$: to obtain feasible solutions;
- secondly the cost $cost(X)$: the main objective;
- finally the deviation from average $DV(X)$.

The algorithm will be given in detail in the following section.

4. The genetic algorithm

Genetic algorithms [8] are part of evolutionary computing, area of artificial intelligence which is rapidly growing. Basically, the GA begins with a set of solutions (represented by chromosomes) called population. Solutions from one population are taken and used to form a new population. This is motivated by a hope, that the new population will be better than the old one. Solutions which are then selected to form new solutions (offspring) are selected according to their fitness – in our case the "roulette wheel" approach [8] is used – the more suitable they are the more chances they have to reproduce. However, unlike the traditional one, in the proposed genetic algorithm mutation and crossover operators are applied alternatively according to a probability Pc.

Also, an "elitist" replacement of the individuals is used here, where the best parent is selected to coexist with the best produced individual (child).

Due to the difficulties to respect the legality and even the feasibility of the solutions generated by the operators of the GA operators, it will be necessary to introduce at each iteration a cooperation between this algorithm and some local search heuristics, i.e. the "legality repair" and "feasibility repair" heuristics.

The following Fig. 2 represents the different steps of the hybrid genetic algorithm process.

In the following subsections, different elements of the proposed GA will be described: the chromosomes encoding in Section 4.1, the initial population building in Section 4.2, the fitness function definition in Section 4.3 and finally the multi-points crossover and the mutation operators will be described in Sections 4.4 and 4.5, respectively. Section 5 will be devoted to the description of the three heuristics.

4.1. The chromosomes

As the solutions have the same form as those from the "crew rostering" problem (i.e. the second subproblem), their representation will be the same [10]. Nevertheless, there is an important difference: here the pairings are not yet obtained as solution of the "crew-pairing" problem (i.e. the first subproblem). So here, it

is necessary to assign any duty periods to the crew members instead of predefined pairings.

One can represent a solution X to the problem by a global schedule of all crew members obtained by the assignment of a set of feasible duty periods to the available crew members.

	Day 1	…	Day j	Day $\|J\|$
Crew 1	-1			
…			-1	
Crew k		-1		
Crew $\|K\|$				-1

A gene x_{kj} takes the value 0 if the crew member k is not assigned to any duty period during the day j, (-1) in the case where the crew member k isn't available during the day j and a positive integer value $x_{kj} = l (l > 0)$ which represents the code associated to the duty period of the day j, belonging to $DP(j)$, assigned to the crew member k.

Let us note by $X = (X_1, \ldots, X_j, \ldots, X_{|J|})$ the $(|J| \times |K|)$ matrix, where X_j is the $(|K| \times 1)$ column vector such that

$$x_{kj} = \begin{cases} l & \text{if the duty period } l \in DP(j) \text{ is assigned to the crew member } k, \\ 0 & \text{if the crew member } k \text{ is not assigned any task during the day } j, \\ -1 & \text{if the crew member } k \text{ is not available to work during the day } j. \end{cases}$$

4.2. Initialization

To build the initial population, of N individuals, one can proceed randomly. However, owing to the diverse restrictions to check, we propose here different ways to construct, as far as possible, a *legal* solution $X_n^0 (n = 1, \ldots, N)$ in the initial population P_0. For this purpose, a subset $\widetilde{DP}(j)$ of "interesting" duty periods must be built.

4.2.1. First step: duty periods determination

The first step consists to determinate, for each day j, a particular subset $\widetilde{DP}(j)$ of duty periods (from all possible ones, i.e. $\widetilde{DP}(j) \subseteq DP(j)$) which will be assigned to different crew members farther.

However, to obtain this subset of duty periods one can proceed according to one of the two following alternatives, namely:

(a) *Initialization based on a linear programming.*

The aim here, is to build a set of duty periods $\widetilde{DP}(j)$ that cover all the flights once. For this purpose, we formulate and solve optimally a set partitioning problem $P(j)$ associated with each day $j \in J$ defined by

$$P(j) \begin{cases} \min z = \sum\limits_{l \in DP(j)} y_{lj} & \\ \sum\limits_{l \in DP(j)} a_{il}^j y_{lj} = 1 & \forall i \in I_j \\ y_{lj} \in \{0, 1\} & \forall l \in DP(j) \end{cases} \quad (1)$$

with the data

$$a_{il}^j = \begin{cases} 1 & \text{if the duty period } l, \text{of the day } j, \\ & \text{contains the flight } i \ (i \in I_{jl}), \\ 0 & \text{otherwise} \end{cases}$$

and the variables

$$y_{lj} = \begin{cases} 1 & \text{if the duty period } l, \text{of the day } j, \text{belongs to the solution,} \\ 0 & \text{otherwise} \end{cases}$$

The constraints (1) ensure that each flight is covered exactly once. The objective function consists in minimizing the number of selected duty periods, i.e. the number of crew members to work during the day j.

The problem $P(j)$ can be solved by means of the CPLEX 9.0 solver to obtain the set of duty periods $\widetilde{DP}(j)$ for each day j.

(b) *Heuristic-based initialization.*

With the same aim, but unlike the precedent approach based on a linear programming, here we proceed heuristically. For this purpose, a flights-based graph $G(j) = (V(j), E(j))$ is build for each day $j \in J$, where the set of nodes $V(j)$ represents the flights and $E(j)$ is the set of directed edges such that an edge (i, i') denotes that the flight i' can follow the flight i in the same duty period (see Fig. 3).

It is evident that each source node in the graph are overlapping flights and must thus be assigned to distinct crew members. Based on this principle, we use the following greedy algorithm to build a set of duty periods $\widetilde{DP}(j)$ that covers all the flights once:

- Consider each source node (indegree 0) separately of the others since each of them must be assigned to different crew members.
- For each source node, select randomly a feasible path according to the applied restriction rules.
- Update the graph $G(j)$ by removing all the nodes that have been used (covered).
- Repeat these steps until all the nodes (flights) are covered.

Then, a set of feasible duty periods, of each day of the month, is obtained. Henceforth, we note by $\widetilde{DP}(j)$ the set of retained duty periods during the day j.

More clearly $\widetilde{DP}(j)$ covers exactly once the flights of the day j, but $\widetilde{DP}(j)$ is not necessary of minimal cardinality.

4.2.2. Second step: duty periods assignment

Afterward and once the set $\widetilde{DP}(j)$ of duty periods is determined for each day j by means of one of the outlined approaches, the obtained duty periods are, assigned to the available crew members in order to build a *legal* solution even if it's not feasible (in the case where the duty periods can not all be assigned). We proceed "day-by-day".

For this purpose, let us note by $DP_k^j(\overline{X}^{j-1})$ the subset of duty periods $l \in \widetilde{DP}(j)$ of the day j that can be assigned to the crew member

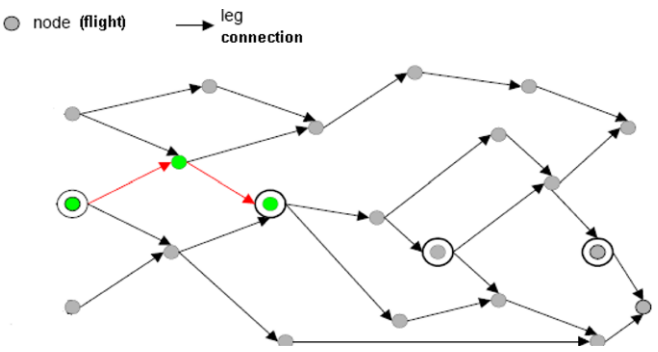

Fig. 3. Flights-based graph.

k, from a *legality* point of view, according to the assignments of all the previous days $j' = 1, \ldots, (j-1)$. ($DP_k^1(\overline{X}^0) \equiv \widetilde{DP}(1)$).

For this purpose, we proceed "pilot-by-pilot" as follows:

(0) *initialization*: put $\overline{K} = \{1, \ldots, K\}; \overline{I}_j = I_j$.
(1) select randomly a crew member $k \in \overline{K}$.
(2) build $DP_k^j(\overline{X}^{(j-1)})$:
- if $DP_k^j(\overline{X}^{(j-1)}) = \emptyset$ then put $x_{kj} = 0$ and go to (4).
- else go to (3).

(3) select a duty period $l \in DP_k^j(\overline{X}^{(j-1)})$ that covers the largest number of flights (if several then select one randomly); put $x_{kj} = l$ and $\overline{I}_j \leftarrow \overline{I}_j \setminus I_{jl}$:
- if $\overline{I}_j = \emptyset$ then stop; put $x_{k'j} = 0\ \forall k' \in \overline{K} \setminus \{k\}$.
- else, put $DP_{k'}^j(\overline{X}^{(j-1)}) \leftarrow DP_{k'}^j(\overline{X}^{(j-1)}) \setminus \{l' | I_{jl'} \cap I_{jl} \neq \emptyset\} \forall k' \in \overline{K} \setminus \{k\}$.

(4) $\overline{K} \leftarrow \overline{K} \setminus \{k\}$:
- if $\overline{K} = \emptyset$ then stop.
- else, return to (1).

4.3. The fitness function

It is clear that even for the population initialization or after the application of the GA operators, the feasibility of the obtained solutions is not guaranteed. For this purpose, we propose here to associate to each individual in the population a fitness function which takes into consideration the case where the solution is feasible or not. Moreover, this fitness function must also take into account the two objectives outlined in Section 3.2.

For this purpose, we defined the fitness function associated to a solution as the linear aggregation of three terms: the penalty term, the total cost (expressed in time) and the deviation function.

So the function to be minimized will be of the form:

$$CT(X) = \beta_1 Pena(X) + \beta_2 cost(X) + DV(X),$$

where the parameters β_1 and β_2 must be defined adequately to minimize hierarchically these three terms, i.e. the penalty first, then the cost and then the deviation function.

The coefficient β_1 must take a value that ensures that $\beta_1 Pena(X) \gg cost(X) \forall X$, but nevertheless these two terms must have comparable scales. Hence, we propose to define the coefficient β_1 as follows.

First of all, for each day $j \in J$ we search for the duty period $l^{(j)} \in DP(j)$ having the largest cost (cf. Section 2.2), i.e. that verifies:

$$CD_{l^{(j)}} = \max_{l \in DP(j)} \{CD_l\}.$$

Then, we build an illegal and non feasible solution in which we suppose that each crew member is assigned to the duty period $l^{(j)}$ for each day j of the month. Here, we suppose that a crew member works during all the days of the month without any free day. Hence the cost of any crew member $k \in K$ will be equal to (cf. Section 2.2):

$$cost_{\max} = \left(\left| \sum_{j \in J} CD_{l^{(j)}} - T_g \right| \right) \times HS.$$

Since we suppose that this cost is generated by every crew member, if we define the coefficient β_1 by $\beta_1 = cost_{\max} \times |K|$, then we can ensure that $\beta_1 Pena(X) \gg cost(X) \forall X$.

As the deviation term $DV(X)$ must be considered only in case of equivalent solutions according to the addition of the two first terms, we defined the coefficient β_2 for each population in the GA, i.e. depending of the different solutions (individuals) X^n inside the same population. Thus, this ensures that the hierarchical order is respected within any population (any iteration) of the GA.

More precisely β_2 must be chosen in such a way that:

$$\begin{cases} \beta_1 Pena(X^n) \geqslant \beta_2 cost(X^n) & \forall n = 1, \ldots, N s.t. Pena(X) \neq 0, \\ \beta_2 cost(X^n) \geqslant DV(X^n) & \forall n = 1, \ldots, N s.t. cost(X) \neq 0 \end{cases}$$

(N is the size of the population)

$$\Rightarrow \begin{cases} \beta_2 \leqslant \min\limits_{n=1,\ldots,Ns.t.cost(X^n)\neq 0} \left\{ \frac{\beta_1 Pena(X^n)}{cost(X^n)} \right\}, \\ \beta_2 \geqslant \max\limits_{n=1,\ldots,Ns.t.cost(X^n)\neq 0} \left\{ \frac{DV(X^n)}{cost(X^n)} \right\}. \end{cases}$$

If we put

$$A = \min_{n=1,\ldots,Ns.t.cost(X^n)\neq 0} \left\{ \frac{\beta_1 Pena(X^n)}{cost(X^n)} \right\}$$

and

$$B = \max_{n=1,\ldots,Ns.t.cost(X^n)\neq 0} \left\{ \frac{DV(X^n)}{cost(X^n)} \right\}$$

we define β_2 by

$$\beta_2 = \begin{cases} \frac{A+B}{2} & \text{if } A \neq 0, \\ B & \text{if } A = 0. \end{cases}$$

Thus, the fitness function proposed in this study takes the form:

$$F(X^n) = \frac{CT_{\max} - CT(X^n)}{CT_{\max}}$$

where $CT_{\max} = \max_{n=1,\ldots,N} \{CT(X^n)\}$ is the largest cost in the current population.

4.4. The multi-points crossover operator

In [10], for the "airline crew rostering" problem, we used a two-points crossover operator to exchange pairings between the two parents solutions. Here, to be able to exchange several duty periods assigned to different crew members, we adopt a multi-points crossover operator.

Let us note by X and Y the two parents selected, by the roulette wheel, from the population and let X' and Y' be the two generated new solutions. We recall that X and Y are "legal" solutions but not necessary "feasible".

We call a gene of a solution, a pair (crew k, day j) of this solution.

The idea of the crossover operator is to exchange a certain number of genes between the solutions X and Y.

We randomly select:

- a number T, with $1 \leqslant T \leqslant \min\{|K|, |J|\}$;
- T distinct genes $(k_t, j_t), t = 1, \ldots, T$ within the solutions X and Y; we have, thus, $k_t \neq k_r$ and $j_t \neq j_r$ for all $t, r \in \{1, \ldots, T\}$ (see Fig. 4).

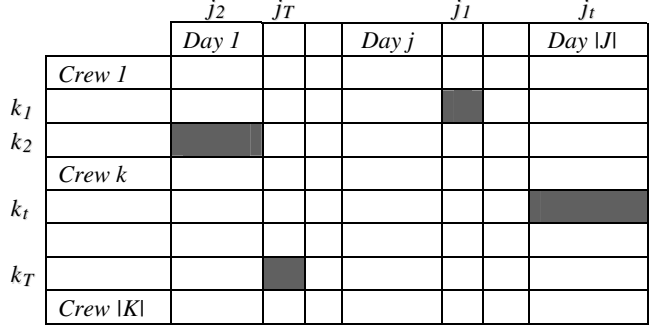

Fig. 4. Selection of T distinct genes.

These selected genes are candidates to be swapped between X and Y. For this purpose, two versions of the crossover operator are proposed.

(a) *Simplified crossover*.

In this simplified version, the exchange of these T genes is achieved automatically in order to obtain the two new solutions X' and Y', i.e. $x'_{k_t j_t} \leftarrow y_{k_t j_t}$ and $y'_{k_t j_t} \leftarrow x_{k_t j_t}$ for all $t = 1, \ldots, T$ and the other genes are not changed.

Clearly, the legality of these two solutions X' and Y' is not assured; for this reason, afterward the "legality repair heuristic" – described farther in this paper (see Section 5.1) – will be applied.

(b) *Probabilistic crossover*.

In this second version, we proceed in two steps.

1. The genes whose content do not violate the legality of any solution X or Y are swapped similarly as in the version a).
2. For the others, the exchange will depend of the degree of non feasibility of the solution, measured by the penalty of the day j_t. More precisely (with, for instance, solution X):
   - if $Pena_{j_t}((X \setminus x_{k_t j_t}) \cup y_{k_t j_t}) \leqslant Pena_{j_t}(X)$, then the switch is accepted, i.e. $x'_{k_t j_t} \leftarrow y_{k_t j_t}$;
   - otherwise, the switch is accepted with a probability P defined by:

$$P = \frac{1}{Pena_{j_t}((X \setminus x_{k_t j_t}) \cup y_{k_t j_t}) - Pena_{j_t}(X) + 1},$$

we have thus

$x'_{k_t j_t} \leftarrow y_{k_t j_t}$ with the probability P;
$x'_{k_t j_t}$ remains unchanged with probability $1 - P$.

This idea is inspired from the principle of the Simulated Annealing with the aim to generate solutions that are not too far from feasibility.

Also, when all the genes $t, t = 1, \ldots, T$ are examined for both solutions, X' and Y' are not necessary legal solutions so the "legality repair heuristic" will be applied to them.

4.5. The mutation operator

Recall that in the proposed genetic algorithm, either the crossover operator is applied, with a probability Pc, or the mutation operator is applied to one of the two selected solutions, with the probability $1 - Pc$.

The proposed mutation operator consists to select, randomly, a day j from a chromosome X, and two distinct crew members k and k' from K such that $x_{kj} \neq -1$ and $x_{k'j} \neq -1$ (see Fig. 5).

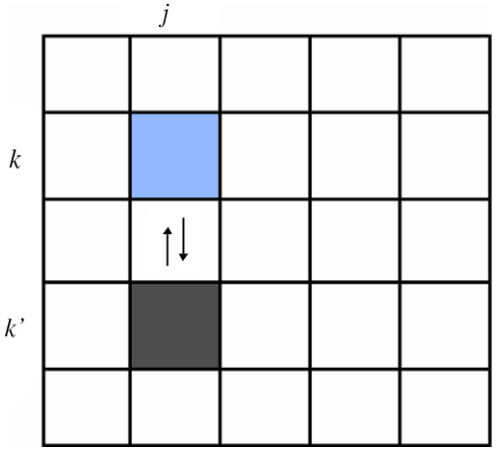

Fig. 5. The mutation operator.

Afterward, the genes x_{kj} and $x_{k'j}$ are switched even if the restriction constraints are violated since this step is immediately followed by the heuristic *Legality Repair* (see Section 5.1).

5. Local search heuristics

In this section, we describe the two local search heuristics: the "Legality Repair Heuristic" (cf. Section 5.1) and the "Feasibility Repair Heuristic" (cf. Section 5.2).

As it is easier to obtain a legal solution that a feasible solution, the first heuristic will always be successful but it will not be the case with the second heuristic. Its aim is only to reduce the unfeasibility of a solution X and this is the reason of the presence of the term $Pena(X)$ in the fitness function (see Section 4.3).

5.1. The "Legality Repair Heuristic"

This heuristic follows immediately the application of the GA operators – crossover or mutation – and is applied to only illegal solutions X, i.e. such that $L(X) \neq \emptyset$; recall that $L(X)$ is the list of genes, i.e. pairs (crew, day), containing duty periods that don't respect all. In fact, due to the GA's operators (see Section 4), for a given solution X it may happen that a duty period $l \in DP(j)$ is assigned to a crew member k without respecting all restrictions. Thus, the aim by this heuristic is to "re-build" a new legal duty period which contains as many flights from the set $I_{lj} \cup ncf_j(X)$ as possible, where $l = x_{kj}$.

The elements of $L(X)$ are examined successively.

Let $(k, j) \in L(X)$ and $x_{kj} = l$. We note by $Nf = I_{lj} \cup ncf_j(X)$ the set of flights to consider (to cover).

(1) build a set $DP_k^j(X)$ of all possible legal duty periods that can fit into the planning of the crew member k and contain, *at least* one flight from *Nf*.
(2) select a duty period $l' \in DP_k^j(X)$ that covers the maximum number of flights from *Nf*; then put $x_{kj} \leftarrow l'$.
(3) if such a duty period does not exist (i.e. $DP_k^j(X) = \emptyset$), then replace the current duty period x_{kj} by a free day, i.e. $x_{kj} \leftarrow 0$.Hence, the legality of any solution is guaranteed at the end of this heuristic.

5.2. The "Feasibility Repair heuristics"

Let us recall here that the feasibility of a solution is satisfied if all the flights are covered one and only one time.

The aim here is to propose an heuristic approach that helps the GA process, by decreasing the penalty of the encountered solutions at each iteration of the GA and improve, thus, their feasibility.

We tested two versions of this heuristic, called "Random Feasibility Repair" and " Improved Feasibility Repair", respectively.

Let X be a non feasible solution. For each flight i – we suppose $i \in I_j$ – we note by r_i the set of duty periods containing the flight i in the solution:

$$r_i = \{l \in DP(j) | i \in I_{jl} \text{ and } \exists k \in Ks.t. x_{kj} = l\}.$$

If $| r_i | = 1$, i.e. the flight is covered exactly once. Thus, only flights such that $| r_i | \neq 1$ are examined.

5.2.1. The "Random Feasibility Repair" heuristic (RFRH)

(1) If $| r_i | = 0$ (the flight i is not covered by the solution) then: select – if there exists at least one – randomly, a crew member k that can be assigned to the flight i, replace the duty period already assigned to k for the day j – i.e. x_{kj} – by another one containing the flight i and, obviously, can fit into the planning of the crew member k (to respect the legality of X).

(2) If $|r_i| > 1$ (the flight i is assigned to more than one crew member and let K_i the set of these crew members); then we proceed as follows:
- select randomly a crew member $\bar{k} \in K_i$;
- leave the flight i in the planning of $\bar{k}$, i.e. do not change $x_{\bar{k}j}$;
- remove the flight i from the planning of all the other crew members $k \in K_i \setminus \{\bar{k}\}$, so
  - If possible, replace x_{kj} by a duty period l that do not contain the flight i ($i \notin I_{jl}$) and obviously can be placed in the planning of k while respecting all the restrictions (the legality of the solution is preserved).
  - otherwise, x_{kj} is set to zero.

5.2.2. The "Improved Feasibility Repair" heuristic (IFRH)

(1) If $|r_i| = 0$ then we proceed as above in Section 5.2.1.
(2) If $|r_i| > 1$ (the flight i is assigned to more than one crew member and let K_i the set of these crew members) then we proceed as follows:
(a) *Initialization*:
- We consider the set $Nf_j(X)$ of interesting flights to cover during the day j:

$$Nf_j(X) = \bigcup_{l \in r_i} I_{jl} \cup ncf_j(X)$$

- For each crew member $k \in K_i$, we determine – by means of a "Depth-first-search" approach – the set $DP^j_k(X)$ of all possible legal duty periods compatible with the roster of the crew member k according to the solution X, i.e. those able to replace x_{kj}.
- For each crew member $k \in K_i$, we determine the set $\overline{DP^j_k}(X)$ of interesting and legal duty periods candidate to replace x_{kj}, i.e. those of $DP^j_k(X)$ containing at least one flight of $Nf_j(X)$:

$$\overline{DP^j_k}(X) = DP^j_k(X) \cap \{l | I_{jl} \cap Nf_j(X) \neq \emptyset\}.$$

Then we consider

$$DP^j(X) = \bigcup_{k \in K_i} \overline{DP^j_k}(X).$$

(b) *Algorithm*: the aim is to assign duty periods to the crew members belonging to K_i such that the flight i is covered exactly once and, also, the largest number of flights from $Nf_j(X)$is covered.
(i) Select a duty period $l \in DP^j(X)$, containing the flight i and covering the largest number of flights: l is such that

$$\begin{cases} i \in I_{jl} \\ |I_{jl}| = \max\limits_{l' \in DP^j(X)} |I_{jl'}| \end{cases} \quad (*)$$

and assign the duty period l to a crew member k, randomly chosen among those such that $l \in \overline{DP^j_k}(X)$: put $x_{kj} \leftarrow l$.
(ii) Updating:

$$K_i \leftarrow K_i \setminus \{k\},$$
$$Nf_j(X) \leftarrow Nf_j(X) \setminus I_{jl},$$
$$DP^j(X) \leftarrow DP^j(X) \setminus l.$$

(iii)
- Stop if $Nf_j(X) = \emptyset$ and put $x_{kj} = 0 \quad \forall k \in K_i$ or if $K_i = \emptyset$.
- otherwise, go to (i) and repeat without considering, anymore, the condition $(*)$ mentioned above since the flight i is already covered.

6. Computational results

At our knowledge, there is no benchmark on which we can test our model, since this problem is a new idea that only few studies have been done on (cf. Section 1). Hence, we applied the proposed model on the real-life datasets (*instance* 1, *instance* 2) used in [10], which deals the problem of assigning a set of pairings to a group of crew members, as well as a new real-life problem (*instance* 3) (these three tests are provided by the airline company "Air-Algérie"):

- *instance* 1: assign 65 pairings, equals to 220 flights, to 5 pilots (of the airplane *B*727) for the period going form 01/03/2004 to 31/03/2004.
- *instance* 2: assign 155 pairings, equivalent to 631 flights, to 19 pilots (of the airplane *AT*72) for the period going from 01/05/2005 to 31/05/2005.
- *instance* 3: A "fresh" problem which consists to assign 558 pairings (equivalent to 1872 flights) to 68 pilots of the (New Generation "NG") airplane (B736 and B738) on the period going from 01/03/2006 to 31/03/2006.

Also, for each of these problems a reference solution, representing the solution proposed by the airline company, is given (see Table 1).

6.1. Best obtained results

Table 1 summarizes the dataset on which our model was applied as well as a comparison between the reference solution proposed by the airline company and the best solution we found.

The second column of the table represents the total number of all generated duty periods of the considered month independently of the crew members availabilities (i.e. $\sum_{j \in J} DP(j)$); the third column indicates the interval within which the number of duty periods generated for the different days of the month varies, and thus, also, the size of the set partitioning problem $P(j)$ (see part (a) of 4.2.1). For instance, for test 1 the number of "daily" duty periods go from 5 to 44 duty periods per day.

We must precise that, five independent runs, of 3000 iterations of the G.A, were made for each test (see Section 6.2). These results were obtained by taking as a probability of crossover $Pc = 0.7$ (after comparison with 0.6, 0.5 and 0.75) for a population of 100 individuals (after comparison with 50 and 150 individuals).

Due to the extent of the proposed model, the outlined solution was obtained after we made a serial of tests and comparison analysis. Hereafter, these analysis will be summarized in the next section.

6.2. Comparative analysis

In this subsection the different alternatives, outlined in the previous sections, will be compared.

Recall that we introduced:

- two versions of the initialization phase: the first is based on a linear programming (see part (a) of Section 4.2.1); the second one is based on a heuristic (see part (b) of Section 4.2.1).These two initializations are examined separately in points 6.2.2 and 6.2.1, respectively.
- Two versions of the crossover operator (see point 4.4) called, respectively, "simplified" and "probabilistic" crossover. The results of these two versions are described in Tables 2 and 4, and Tables 3 and 5, respectively.
- Two versions of the "feasibility repair heuristic" (see points 5.2.1 and 5.2.2) called, respectively, "Random Feasibility Repair

Table 1
Dataset

Tests	# Duty periods $\left(\sum_{j\in J}\lvert DP(j)\rvert\right)$	# Columns $(DP(j))$	Reference solution X		Our model's best solution $\bar{X}$		
			$cost(X)$	$DV(X)$	$cost(\bar{X})$	$DV(\bar{X})$	CPU time $sc \setminus iter$
instance 1	595	$5 \leqslant Col. \leqslant 44$	450	75:26	0	00:08	0.10
instance 2	2333	$28 \leqslant Col. \leqslant 121$	0	272:04	0	00:50	0.43
instance 3	17485	$278 \leqslant Col. \leqslant 1382$	0	927:17	0	211:35	2.33

Table 2
Simplified crossover

Tests	"RFRH"			"IFRH"			"RFRH" & "IFRH"			CPU time
	Best cost	Best dev.	Aver. dev.	Best cost	Best dev.	Aver. dev.	Best cost	Best dev.	Aver. dev.	
instance 1	0	00:08	00:19	0	00:08	00:21	0	00:08	00:20	5 minutes 0
instance 2	0	01:22	01:29	0	00:52	01:20	0	01:00	01:11	21 minutes 30
instance 3	X	X	X	0	248:31	254:29	0	231:03	242:09	1 hour 56

Table 3
Probabilistic crossover

Tests	"RFRH"			"IFRH"			"RFRH" & "IFRH"			CPU time
	Best cost	Best dev.	Aver. dev.	Best cost	Best dev.	Aver. dev.	Best cost	Best dev.	Aver. dev.	
instance 1	0	00:08	00:16	0	00:08	00:15	0	00:08	00:14	5 minutes 30
instance 2	0	01:04	01:08	0	00:50	01:06	0	01:02	01:28	22 minutes 0
instance 3	X	X	X	0	230:15	243:44	0	230:41	234:52	1 hour 59

Heuristic" (RFRH) and "Improved Feasibility Repair Heuristic"(IFRH). The obtained results are presented in Tables 2–5.

6.2.1. The heuristic-based initialization

Recall (see part (a) of 4.2.1) that in this part, a flights-based graph is built, first, then a set of legal duty periods is randomly determined for each day of the month. These duty periods are then assigned to the available crew members while building legal pairings and rosters.

Probabilistic vs. simplified crossover. Hereafter, a comparison between the results obtained when we used the simplified crossover against the probabilistic one:

In Tables 2 and 3 – and forthcoming Tables 4 and 5 – columns 2, 3 and 4 (columns 5, 6 and 7, respectively) give the cost $(cost(X))$, the deviation function $(DV(X))$ and the average value of the deviations given by the five runs and obtained with the application of the (RFRH) (IFRH, respectively).

The same results are given in columns 8, 9 and 10 but, this time, when (RFRH) and (IFRH) are used successively: with a probability 0.5 RFRH (IFRH) is applied to both new individuals.

Finally, column 11 provides the total CPU time (on a Pentium IV, 2.5 GHz, 256 SDRAM) which is similar for each alternative.

It appears from these two tables

- That with RFRH we do not generate any feasible solution for the instance 3, either applying the simplified or probabilistic crossover.
- For all the other tests, and on the five independent runs, the obtained solutions generate no cost (that is why it do not exist any column "average cost" in Tables 2 and 3).
- For instance 1, we obtain with each alternative the same best solution (which is probably the optimal one).
- For instances 2 and 3, "IFRH" and "RFRH&IFRH" give similar results with a slight advantage for the second one, from the

Table 4
Simplified crossover

Tests	"RFRH"			"IFRH"			"RFRH" & "IFRH"			CPU time
	Best cost	Best dev.	Aver. dev.	Best cost	Best dev.	Aver. dev.	Best cost	Best dev.	Aver. dev.	
instance 1	0	00:08	00:21	0	00:08	00:20	0	00:08	00:24	4 minutes 30
instance 2	0	02:12	02:12	0	01:10	01:50	0	01:12	01:56	18 minutes 30
instance 3	X	X	X	0	223:55	226:15	0	220:09	227:34	1 hour 48

Table 5
Probabilistic crossover

Tests	"RFRH"			"IFRH"			"RFRH" & "IFRH"			CPU time
	Best cost	Best dev.	Aver. dev.	Best cost	Best dev.	Aver. dev.	Best cost	Best dev.	Aver. dev.	
instance 1	0	00:08	00:19	0	00:08	00:19	0	00:08	00:26	4 minutes 30
instance 2	0	01:22	01:38	0	00:58	01:31	0	01:22	01:41	17 minutes 30
instance 3	X	X	X	0	211:35	224:01	0	214:05	218:19	1 hour 40

point of view of the "average deviation". Nevertheless, the best results are always obtained with "IFRH".
- The probabilistic crossover is slightly better than those obtained with the simplified crossover.

6.2.2. The linear programming-based initialization

Probabilistic vs. simplified crossover. First of all, the same observations of the preceding subsection can be given here, on basis of Tables 4 and 5.

If we compare, now, the two proposed initializations each other, the linear programming-based initialization appears better than the heuristic-based one for instance 3, so when the size of the problem become larger.

7. Conclusion

In the present paper, we proposed a new and efficient model that enable us to formulate and solve the airline crew-pairing and rostering problems simultaneously. The proposed resolution approach is, principally, based on a hybrid genetic algorithm.

Nevertheless, many alternatives of this GA are proposed and compared. In fact, a heuristic-based initialization is compared to a linear programming-based one and, also, a probabilistic multi-points crossover is compared to a simplified one. Moreover, and with the aim to improve the feasibility of the obtained solutions, two local search heuristics ("RFRH" and "IFRH") are combined to the GA, separately first, and simultaneously, then.

Roughly speaking, the outlined results (Tables 2–5) confirmed that the proposed model succeeds to solve the three, real-world, tests in a short computing time knowing that the proposed approach tackles the airline crew-pairing and rostering problems simultaneously.

More precisely, and with regards to the comparison of the different versions of the hybrid GA, one can conclude the following:

- Generally, the probabilistic multi-points crossover gives more interesting value of the deviation (the best and average deviation) compared with the simplified one probably because it provides more diversification since it's inspired from the simulated Annealing principle.
- The, heuristic "IFRH" is more efficient than the "RFRH" and especially for the large size instance *instance* 3 where the "RFRH" failed to find any feasible solution on the 5 runs.
- The linear programming initialization is the best proposed alternative compared to the heuristic-based one, because, not only, it provides the best deviation, but also, the first feasible solutions were obtained after a few number of GA iterations: less than 10 iterations for *instance* 1, less than 40 iterations for *instance* 2 and less than 60 iterations for *instance* 3. This can be explained by the fact the best set of duty periods covering all the flights once are determined a priori.
- Finally, we have to precise that the best solutions of the previous tables were, also, obtained early in the G.A (less than 500 iterations for instance 1, less than 1000 iterations for instance 2 and less than 2400 iterations for instance 3). Thus, the 3000 iterations is the number of the GA iterations beyond which the best found solution don't change.

An interesting question is the comparison between the integrated approach described in this paper and the classical non-integrated approach, i.e. the successive resolution of the two subproblems. Three facts result of a first comparison [9] based only on the three instances of Section 6:

- the GA proposed in [9] is unable to solve the "airline crew pairing" of instance 3 due to a too large number of possible pairings; thus we were unable to consider the "airline crew rostering";
- for the instance 2, if only the reduced set of "optimal pairings" resulting of the first subproblem, is used, the second subproblem appears impossible;
- for the instance 1, the solution obtained with the integrated approach (see Table 1) is better than the one obtained with the non-integrated approach, corresponding to a cost of zero, but an average deviation of 49:28.

Clearly, such comparison between integrated and non-integrated approaches merits more attention and larger experimentations in the future.

References

[1] B. Gopalakrishnan, E.L. Johnson, Airline crew scheduling: State-of-the-Art, Annals of Operations Research 140 (2005) 305–337.
[2] Y. Guo, T. Mellouli, L. Suhl, M. Thiel, A partially integrate airline crew scheduling approach with time-dependent crew capacities an multiple home bases, European Journal of Operational Research 171 (3) (2006) 1169–1181.
[3] D. Levine, Parallel genetic algorithms for the set partitioning problem, PhD thesis, Argonne National Laboratory, Maths and Computer Science Division, May 1994, Report ANL 94/23.
[4] M. Minoux, F.M. Zeghal, Modeling and solving the crew assignment problem in air transportation, European Journal of Operational Research 175 (1) (2006) 187–209.
[6] A. Mercier, F. Soumis, An integrate aircraft routing, crew scheduling and flight retiming model, Computers and Operations Research 34 (8) (2007) 2251–2265.
[7] Shaw Ching Chang, A new aircrew scheduling model for short-haul routes, Journal of Air Transportation Management 8 (2002) 249–260.
[8] F. Glover, G.A. Kochenberger (Eds.), Handbook of Metaheuristics, Operations Research Management Science, Kluwer Academic Publishers, 2003.
[9] N. Souai, Modeling and solving the airline crew scheduling problem by hybrid metaheuristics, PhD thesis. Laboratory of Mathematics and Operations Research, Faculty of Engineering, Mons, Belgium, February 2007.
[10] N. Souai, J. Teghem, Hybridizing the Genetic Algorithm and the Simulated Annealing for the Airline Crew Rostering Problem, ValenSciences No. 5, Press Universitaires de Valenciennes, 2006. pp. 323–344.