Expert Systems with Applications 40 (2013) 7504–7512

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

A Memetic Algorithm for staff scheduling problem in airport security service ☆

Anas Abdoul Soukour [a,b], Laure Devendeville [b,*], Corinne Lucet [b], Aziz Moukrim [c]

[a] DSI, ICTS France, Roissypole le dôme, 1 rue de la Haye, BP 12936, 95732 Roissy CDG Cedex, France
[b] Laboratoire MIS – EA 4290/Université de Picardie Jules Verne, 33 rue Saint Leu, 80039 Amiens Cedex 1, France
[c] Laboratoire HeuDiaSyC – CNRS UMR 7253/Université de Technologie de Compiègne, Centre de Recherches de Royallieu, BP 20529, 60205 Compiègne Cedex, France

ARTICLE INFO

Keywords:
Planning
Scheduling algorithms
Memetic Algorithm

ABSTRACT

The staff scheduling problem is widely studied in Operational Research. Various surveys are available in the literature dealing with this problem which concerns various objectives and various constraints. In this article, we present a staff scheduling problem in airport security service. First, a modeling of the problem, and a representation of solutions are shown. The problem is solved in three steps, days-off scheduling, shift scheduling, and staff assignment. We focus on the last step, by providing a Memetic Algorithm (*MA*) which merged an Evolutionary Algorithm and Local Search techniques. We propose a chromosome encoding, a crossover operator and a combined neighborhood function, specially dedicated to this staff assignment problem. Besides providing better solutions than software currently used, this algorithm provides up to 50% of improvement from initial feasible solutions.

1. Introduction

This paper presents a scheduling problem in airport security service. In this particular application area, human workforce is spread to control various check points on airport ground, for airlines and airport operators. Activities consist in controlling and/or supervising strategic points of a particular geographical area involving a large number of persons either moving (tarmac, baggage area, etc.) or remaining in a location (check-in, boarding area, etc.) for a certain time. Scheduling these activities, distributed mostly to large geographical areas, is done according to legal constraints, preferences of employees and customer's requirements. One of the reasons of the problem complexity is due to large quantity of employees working in different places. Employment contracts are various and numerous. Moreover they are specified with multiple legal regulations, due to French law, and collective agreements. In addition, there are many kinds of skills with different levels. Moreover, employee requirement expressed by airport operators or airlines is characterized by a large fluctuation in the horizon time.

From a complexity point of view, the Timetable Design problem is NP-Complete (Brucker, Qu, & Burke, 2011; Garey & Johnson, 1979). Various surveys are available in literature dealing with scheduling problem (Ásgeirsson, 2012; Blöchliger, 2004; Ernst, Jiang, Krishamoorthy, & Sier, 2004; Genç, Erol, Eksin, Berber, & Güleryüz, 2012; Günther & Nissen, 2012).

Solving staff scheduling problem is generally decomposed in three or four steps, for complexity reasons (Detienne, Péridy, Pinson, & Rivreau, 2009), but also in order to include data depending on feedback experiences. The first step, *days-off scheduling*, consists in specifying a sequence of days of work and days-off for each employee according to requirement of number of employees. It is constrained by employee's contract (for example working only weekends or specific days of week) or collective agreements (for example number of weekends off). The second step consists in selecting, from a potentially large pool of shifts, what shifts are to be worked, and how many are needed in order to meet demand. For this *shift scheduling* problem, two kinds of models exist in literature: task covering model, based on Dantzig model (Soumis, Pesant, & Rousseau, 2005), implicit models defined by Rekik, Cordeau, and Soumis (2004), Aykin (1998) or also Bechtold and Jacobs models (Bechtold & Jacobs, 1990). The last step, *staff assignment*, consists in assigning shifts, defined in step two, to employees, with the best satisfaction, and with respect of rules and regulations. Data from staff scheduling problem are various and numerous. Each combination of them provides a particular prob-

☆ This paper is an extension of a preliminary work published in 14th IFAC Symposium on Information Control Problems in Manufacturing proceedings (Abdoul Soukour, Devendeville, & Lucet, 2012).
* Corresponding author. Tel.: +33 3 22 82 59 09.
E-mail addresses: anas.abdsouk@ictsfrance.fr (A. Abdoul Soukour), laure.devendeville@u-picardie.fr (L. Devendeville), corinne.lucet@u-picardie.fr (C. Lucet), aziz.moukrim@utc.fr (A. Moukrim).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.06.073

lem which is commonly solved, in industrial context, by heuristic methods or metaheuristics.

Our problem was decomposed into these three steps previously presented. In this paper we deal with the last step, staff assignment, using data provided by the current system used by planning operators. We propose a metaheuristic based on a Memetic Algorithm. Memetic Algorithms (*MA*) are classified as Evolutionary Algorithms and are first introduced by Moscato (1999) and several studies showed its efficiency (Bouly, Dang, & Moukrim, 2010). It consists in a combination of an evolutionary process with Local Search (LS) methods to improve the population.

In Section 2, we first show an integer programming modeling of the problem that allows to clearly express all the constraints. We also define and present solutions with their associated operators. The criteria we used to evaluate the quality of a solution are detailed. Section 3 develops our Memetic Algorithm approach for the assignment problem. An efficient chromosome encoding, a crossover operator and a combined neighborhood function are shown. We also propose some greedy algorithms, used to initialize the population. Numerical results on real instances and their analysis are presented in Section 4. Then conclusions and perspectives end this article.

2. Problem modeling and construction of a feasible solution

For a given time period, the horizon, we have to satisfy demand expressed by a number of employees required in time unit that we call *workforce demand*. This demand is defined by clients and transformed by planning operators in a set of *shifts*. To fulfill this demand, the company has a fixed number of employees. To each employee is associated a plan or a cycle, specifying the workdays and the days-off. Our aim is to assign the set of shifts to available employees during this period, optimizing a cost function.

2.1. Data

A *shift* represents a security activity of an employee, for a day of the horizon. It is defined by starting and ending times, required skill and geographical area. It also includes meal break. Overall, shifts are defined according to working laws, especially controlling the respect of maximal and minimal legal hours. A shift may contain different tasks requiring different skills and possibly on different geographical areas. Travel time between two areas is considered when making shifts. In a general way, skills are hierarchically organized. When multi-skill is required, skill with upper level is selected as required skill for the considered shift. As an example, Fig. 1 shows a shift that starts at 8:00 and finishes at 16:00. Its associated skill is SKILL A, because of the higher level of SKILL A in the skill hierarchy and it is concerned by the two areas 1 and 2.

In order to satisfy workforce demand, we have a set of *employees*. Each employee is characterized by a type of contract that fixes his monthly working time, a plan (cycle) of workdays and days-off through the horizon, his higher level skill and some personal restrictions. In some case, medical restrictions prevent some employee from assignment on some geographical areas. Some employees work only on week-end and others never on Wednesday. Each employee is assigned a starting time before which he cannot start working and an end time beyond which he cannot complete its work. An employee is *over time scheduled* if the sum of his assigned shifts during the horizon is greater than this contractual working time. He is *under time scheduled* otherwise.

Fig. 2 shows an example of employee characteristics and his plan for an horizon of 30 days.

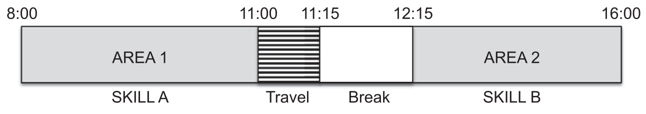

Fig. 1. Shift example.

2.2. Modelisation

To provide a more formal explanation of the problem we have included an integer-programming modeling of our specific staff scheduling problem in this section. We present the main constraints on working laws and assignment consistency, and also the main criteria of cost evaluation.

Data:

Q	set of shifts $q \in Q$
J	set of days $j \in J$
E	set of employees $e \in E$
QJ_j	set of shifts belonging to day j, $QJ_j \subseteq Q$
QE_e	set of shifts matching with skill, availability and area of employee e, $QE_e \subseteq Q$
EQ_q	set of employees matching with skill, schedule and area of shift q, $EQ_q \subseteq E$
ω_q	starting time of shift q
v_q	ending time of shift q
NH_e	number of contractual hours per month for employee e
$Start_{week}$	set of first day of weeks
MWD	maximum number of consecutive working days
MH	maximum number of hours between two working shifts
MHW	maximum number of working hours for a week
$\$_q$	cost of the non assignment of shift q
$\$sup_e$	cost of an extra working hour for employee e
$\$inf_e$	cost of missing working hour (under contract) for employee e

Variable:

$x_q^e = 1$ if shift q is assigned to employee e

Minimize function:

$$\$UnderC + \$OverT + \$UnderT + \$EmpD \tag{1}$$

Under constraints:

$$\forall e \in E, \quad \forall j \in J, \quad \sum_{q \in QJ_j \cap QE_e} x_q^e \leqslant 1 \tag{2}$$

$$\forall q \in Q, \quad \sum_{e \in EQ_q} x_q^e \leqslant 1 \tag{3}$$

$$\forall e \in E, \quad \forall j \in [1, |J| - MWD], \quad \sum_{j'=j}^{j+MWD} \sum_{q \in QJ_{j'} \cap QE_e} x_q^e \leqslant MWD \tag{4}$$

$$\forall e \in E, \quad \forall j \in [1, |J| - 1], \quad \left(\sum_{q \in QJ_j \cap QE_e} x_q^e \right) \cdot \left(\sum_{q \in QJ_{j+1} \cap QE_e} x_q^e \right) \cdot \left[\sum_{q \in QJ_{j+1} \cap QE_e} x_q^e \cdot \omega_q - \sum_{q \in QJ_j \cap QE_e} x_q^e \cdot v_q - MH \right] \geqslant 0 \tag{5}$$

$$\forall e \in E, \quad \forall j \in Start_{week} \sum_{j'=j}^{j+6} \sum_{q \in QJ_{j'} \cap QE_e} x_q^e \cdot (v_q - \omega_q) \leqslant MHW \tag{6}$$

First, to reduce the number of variables, we compute sets QJ_j, QE_e and EQ_q. Eqs. (2) and (3) ensure that an employee does not work more that one shift a day and that every shift is assigned to

Employee X	SKILL	Type of Contract	Start time	End time	Restriction
	A	145 hours	8:00	22:00	no

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
	off	off						off	off	off					off	off								off	off	off				

Fig. 2. Plan example for employee X.

a single employee. Constraints (4)–(6) relate to the laws that prohibit for each employee, more than *MWD* consecutive working days (4), less than *MH* hours between two shifts (5) and more than *MHW* hours of work for a week (6). The objective function takes into account the following costs:

- *Under coverage:* $*UnderC*. The cost of under coverage is established for non-assigned shifts. These costs are computed according to the shift duration.

$$\$UnderC = \sum_{q\in Q}\left(1 - \sum_{e\in EQ_q} x_q^e\right).\$_q \tag{7}$$

- *Over time scheduling:* $*OverT*. The cost of over time scheduling is expressed by the sum of costs related to scheduled working hours above the contractual working hours NH_e for every employee *e*.

$$\$OverT = \sum_{e\in E}\max\left(0, \left(\sum_{q\in QE_e} x_q^e.(v_q - \omega_q) - NH_e\right)\right)\cdot \$sup_e \tag{8}$$

- *Under time scheduling:* $*UnderT*. The cost of under time scheduling is expressed by the sum of cost related to remaining working hours to reach the contractual working hours.

$$\$UnderT = \sum_{e\in E}\max\left(0, \left(NH_e - \sum_{q\in QE_e} x_q^e\cdot(v_q - \omega_q)\right)\right)\cdot \$inf_e \tag{9}$$

- *Employee dissatisfaction:* $*EmpD*. Some social criteria affect the evaluation of the solution, such as allowing employees to be free on weekends, avoiding isolated days-on or days-off. An *isolated day-on* is a working day followed and preceded by at least one day-off. An *isolated day-off* is an off day followed and preceded by at least one working day. This cost cannot be more detailed for confidential reasons.

2.3. Solution and operator associated

A solution for our problem is defined as the assignment of shifts to employees on the horizon. This solution is defined by a set of couples (shift, employee) but also by non-assigned shifts.

Definition 1. Formally a solution S is defined as follows: $S = \langle \mathcal{A}, Q_{AV}, E\rangle$ where $\mathcal{A} \subset Q \times E$, Q is the set of shifts, E is the set of employees and Q_{AV} is the set of non-assigned shifts: $Q_{AV} = \{q \in Q / \forall e \in E, (q,e) \notin \mathcal{A}\}$.

Obviously, all solutions considered in this work have to respect work laws and conditions of effective organization described by the constraints in Section 2.2.

Definition 2. A solution $S = \langle \mathcal{A}, Q_{AV}, E\rangle$ is said **feasible** if the constraints (2)–(6) in Section 2.2 are respected.

In order to build a solution, some operators are needed:

Definition 3 ($\oplus$ *operator*). Let $S = \langle \mathcal{A}, Q_{AV}, E\rangle$ be a feasible solution and $(q,e) \in Q_{AV} \times E$ then $S \oplus (q,e) = \langle \mathcal{A} \cup \{(q,e)\}, Q_{AV} \setminus \{q\}, E\rangle$. Moreover, $\oplus$ operator has identity elements (q,ϵ) such that $q \in Q$ and ϵ is a fictive employee ($\epsilon \notin E$): $\forall q \in Q$, $S \oplus (q,\epsilon) = S$.

This operator has some interesting properties which we shall refer to further on the presentation of our method.

Property 1. *Let $S = \langle \mathcal{A}, Q_{AV}, E\rangle$ be a feasible solution, $(q_1,e_1) \in Q_{AV} \times E$, $(q_2,e_2) \in Q_{AV} \times E$ with $q_1 \neq q_2$, $S \oplus (q_1,e_1)$ and $S \oplus (q_2,e_2)$ be two feasible solutions. If $e_1 \neq e_2$, $(S \oplus (q_1,e_1)) \oplus (q_2,e_2)$ is feasible too. Otherwise, $(S \oplus (q_1,e_1)) \oplus (q_2,e_1)$ is feasible iff q_1 and q_2 do not overlap in time and the constraints (4)–(6) in Section 2.2 are not violated for e_1. Moreover, in both cases $(S \oplus (q_1,e_1)) \oplus (q_2,e_2) = (S \oplus (q_2,e_2)) \oplus (q_1,e_1)$.*

Definition 4 ($\bigoplus$ *operator*). Let $S = \langle \mathcal{A}, Q_{AV}, E\rangle$ be a feasible solution and $\mathcal{L} = \{(q_1,e_1), \ldots (q_n,e_n)\}$ where $\forall i \in [1,n]$, $(q_i,e_i) \in Q_{AV} \times E$ such that $\forall j \in [1,n]$ with $i \neq j$, $q_i \neq q_j$. We denote $S \bigoplus \mathcal{L} = S_n$ where $S_0 = S$ and $\forall k \in [1,n]$, $S_k = S_{k-1} \oplus (q_k,e_k)$.

Property 2. *If $S \oplus (q_1,e_1), \ldots, S \oplus (q_n,e_n)$ are feasible solutions then: If $\forall i \in [1,n]$, $j \in [1,n]$ with $i \neq j$, such that $e_1 \neq e_j$, by extension of* Property 1 *$S \bigoplus \mathcal{L}$ is feasible too. Otherwise $S \bigoplus \mathcal{L}$ is feasible iff all shifts $q_i \in Q_e$ where $Q_e = \{q_i / (q_i,e) \in \mathcal{L}\}$, do not overlap in time and the constraints* (4)–(6) *in Section* 2.2 *are not violated for e.*

Definition 5 ($\ominus$ *operator*). Let $S = \langle \mathcal{A}, Q_{AV}, E\rangle$ and $(q,e) \in \mathcal{A}$ then $S \ominus (q,e) = \langle \mathcal{A} \setminus \{(q,e)\}, Q_{AV} \cup \{q\}, E\rangle$.

Definition 6 ($\bigominus$ *operator*). Let $S = \langle \mathcal{A}, Q_{AV}, E\rangle$ be a feasible solution and $\mathcal{L} = \{(q_1,e_1), \ldots (q_n,e_n)\}$ where $\forall i \in [1,n]$, $(q_i,e_i) \in Q_{AV} \times E$ such that $\forall j \in [1,n]$ with $i \neq j$, $q_i \neq q_j$. We denote $S \bigominus \mathcal{L} = S'_n$ where $S'_0 = S$ and $\forall k \in [1,n]$, $S'_k = S'_{k-1} \ominus (q_k,e_k)$.

Properties 1 and 2 can be extended to operators $\ominus$ and $\bigominus$.

2.4. Quality of a solution

There are many solutions, but not all of them have the same implication from a quality point of view. The quality of solution S is defined by the objective function $f(S)$ (see Eq. (10)).

$$f(S) = w_1 \cdot \$UnderC(S) + w_2 \cdot \$OverT(S) + w_3 \cdot \$UnderT(S) + w_4 \cdot \$EmpD(S) \tag{10}$$

Terms $*UnderC*(*S*), $*OverT*(*S*), $*UnderT*(*S*) and $*EmpD*(*S*) refer to costs defined in Eq. (1) applied to solution *S*. Weights w_k are used to promote the quantification of terms and are established according to planning operator expertise. Solving our problem corresponds to finding solution S such as $f(S)$ is minimal. Costs of criteria are on the same scale and expressed in homogeneous unit (HU).

3. Memetic Algorithm

As a first step, we developed greedy algorithms and Local Search methods to solve the problem. To improve these first results, we have worked out a Memetic Algorithm. Such kind of metaheuristic is recent (Moscato, 1999). It combines Genetic Algorithm and Local Search techniques. The specific concepts of Genetic Algorithms are introduced in Holland (1992). A population of solutions (individuals) evolves from a generation to another. The individuals are encoded into a similar structure called a *chromosome*. The individuals of the current generation are combined by a *crossover* operator to produce the individuals of the next one. A diversification process is also used to avoid homogeneity in the population. This diversification is obtained through a *mutation* operation. Local Search steps in this process regarding Memetic Algorithm, not only to diversify but also to improve the current population. To insert an individual into the next population and to identify improvements, it is necessary to know the performance of each individual, its *fitness*, in the population through an evaluation procedure. In our study, this evaluation involves the objective function f in Eq. (10) presented in Section 2.4.

3.1. Chromosome encoding

To encode solutions, we consider a fixed order of shifts in Q, denoted $Q_0 = \langle q_1, q_2, \ldots, q_{|Q|}\rangle$. Let $S_k = \langle A, Q_{AV}, E\rangle$ be a feasible solution, then S_k is encoded by chromosome $C_k = \langle (q_1, e_1), (q_2, e_2), \ldots, (q_{|Q|}, e_{|Q|})\rangle$, respecting Q_0 order, such that e_j is an employee of E if the assignment $(q_j, e_j) \in A$ and $e_j = \epsilon$, the fictive employee, otherwise ($q_j \in Q_{AV}$ in that case). Obviously, e_i and e_j with $i \neq j$ could be the same employee, because one employee may work more than once on the horizon. We notice that for any two different chromosomes the sequences of the shifts are always the same. So, we can reduce the expression of C_k to the sequence of employees $\langle e_1, e_2, \ldots, e_{|Q|}\rangle$. We will denote $C_k[i] = e_i$ the ith *gene* of the chromosome C_k. At each solution S_k corresponds a chromosome C_k ($C_k = Dual(S_k)$) and vice versa ($S_k = Dual(C_k)$), and the fitness of C_k is $f(S_k)$. We notice that our encoding is a *direct* one, because chromosomes fully describe solutions.

Example 1. As an illustration, solution

$$S_1 = \langle\{(q_1, b), (q_3, a), (q_4, b)\}, \{q_2\}, \{a, b\}\rangle$$

is represented by chromosome $C_1 = \langle b, \epsilon, a, b\rangle$.

3.2. Selection operator

In order to provide a new generation, we need to recombine some chromosomes. In our method we chose to elect two chromosomes to create a new one. There are many ways to choose the parents (roulette-wheel selection, truncation selection, tournament selection, etc.). In this work, we use the tournament selection (Prins, 2004). k individuals are randomly selected and the one with the best fitness is retained as first parent, chromosome C_1. The same applies for the second parent, chromosome C_2. Objective is to transmit good properties of parents to offspring. These parents must be different.

Algorithm 1. Crossover

```
Input:
  $Q$: set of available shifts
  $E$: set of available employees
  $C_1$: chromosome
  $C_2$: chromosome
  $X$: random position in chromosome $[1, |Q|]$
Output:
  $C_{neo}$: chromosome

  $C_{neo} \leftarrow \langle \epsilon, \epsilon, \ldots, \epsilon\rangle$
  $S_{neo} \leftarrow Dual(C_{neo})$
  for $r = 0, \ldots, |Q| - 1$ do
    $i \leftarrow ((r + (X - 1)) modulo |Q|) + 1$
    if $S_{neo} \oplus (q_i, C_1[i])$ is feasible then
      if $S_{neo} \oplus (q_i, C_2[i])$ is feasible then
        if $f(S_{neo} \oplus (q_i, C_1[i])) \leqslant f(S_{neo} \oplus (q_i, C_2[i]))$ then
          $C_{neo}[i] \leftarrow C_1[i]$
          $S_{neo} \leftarrow S_{neo} \oplus (q_i, C_1[i])$
        else
          $C_{neo}[i] \leftarrow C_2[i]$
          $S_{neo} \leftarrow S_{neo} \oplus (q_i, C_2[i])$
        end if
      else
        $C_{neo}[i] \leftarrow C_1[i]$
        $S_{neo} \leftarrow S_{neo} \oplus (q_i, C_1[i])$
      end if
    else
      if $S_{neo} \oplus (q_i, C_2[i])$ is feasible then
        $C_{neo}[i] \leftarrow C_2[i]$
        $S_{neo} \leftarrow S_{neo} \oplus (q_i, C_2[i])$
      end if
    end if
  end for
  return $C_{neo}$
```

3.3. Crossover operator

In order to produce new solutions we use Crossover Operator by first selecting two parents among the population and constructing one new individual by combining their genes. Parents are chosen as described previously. There exists numerous types of crossover operator in the literature, depending on the considered problems like the Traveling Saleman Problem (Oliver, Smith, & Holland, 1987) or the Scheduling problems (Portmann & Vignier, 2000) and obviously depending on their encoding. We used an operator adapted to our problem that is a parallel version of the well known 1-point crossover operator. The main idea is to compare the ith gene of the first parent with the ith gene of the second parent and transmit the best one to the offspring, starting from a random position in the chromosomes.

Let C_1 and C_2 be the parent chromosomes and C_{neo} be their child under construction, we note $C_{neo} = Crossover(C_1, C_2)$. Initially $C_{neo} = \langle \epsilon, \epsilon, \ldots, \epsilon\rangle$ and the associated solution is $S_{neo} = \langle \emptyset, Q, A\rangle$. Let X be a random value in $[1, |Q|]$. The genes of the same rank of C_1 and C_2 are compared from the X-th rank in a circular manner. The best of both genes, subject to feasibility, is transmitted to C_{neo}. Algorithm 1 gives the details of the construction of C_{neo} by our crossover operator. The complexity of this algorithm is $O(|Q|)$.

Example 2. For example, let $Q = \{q_1, q_2, q_3\}$ be the set of available shifts, $E = \{a, b\}$ be the set of available employees, $C_1 = \langle a, \epsilon, a\rangle$ and $C_2 = \langle b, a, \epsilon\rangle$ be the parent chromosomes. Suppose that $X = 2$.

Initially $C_{neo} = \langle \epsilon, \epsilon, \epsilon \rangle$ and $S_{neo} = \langle \emptyset, \{q_1, q_2, q_3\}, \{a, b\} \rangle$.

1. First we compare $S_{neo} \oplus (q_2, C_1[2])$ and $S_{neo} \oplus (q_2, C_2[2])$, suppose that both are feasible and $f(S_{neo} \oplus (q_2, C_1[2])) > f(S_{neo} \oplus (q_2, C_2[2]))$, then $C_{neo} = \langle \epsilon, a, \epsilon \rangle$ and $S_{neo} = \langle \{(q_2, a)\}, \{q_1, q_3\}, E \rangle$
2. Next we compare $S_{neo} \oplus (q_3, C_1[3])$ and $S_{neo} \oplus (q_3, C_2[3])$, suppose that only $S_{neo} \oplus (q_3, C_2[3])$ is feasible, then $C_{neo} = \langle \epsilon, a, \epsilon \rangle$ and $S_{neo} = \langle \{(q_2, a)\}, \{q_1, q_3\}, E \rangle$
3. Last, we compare $S_{neo} \oplus (q_1, C_1[1])$ and $S_{neo} \oplus (q_1, C_2[1])$, suppose that both are feasible, moreover $f(S_{neo} \oplus (q_1, C_1[1])) \langle f(S_{neo} \oplus (q_1, C_2[1]))$ then $C_{neo} = \langle a, a, \epsilon \rangle$ and $S_{neo} = \langle \{(q_1, a), (q_2, a)\}, \{q_3\}, E \rangle$

So the new chromosome is $C_{neo} = \langle a, a, \epsilon \rangle$.

Algorithm 2. IDCshifts

```
Input:
  $S_0$: initial solution
  $D_{max}$: number of assignments to destroy
Output:
  $S_{best}$: best solution
  $S \leftarrow S_0$
  $S_{best} \leftarrow S$
  repeat
    $d \leftarrow$ random choice between 1 and $D_{max}$
    $S^d_{dest} \leftarrow Destruction(S, d)$
    if $f\left(S^d_{dest}\right) < f(S)$ then
      $S_{best} \leftarrow S^d_{dest}$
    end if
    $S_{const} \leftarrow Construction\left(S^d_{dest}\right)$
    if $f(S_{const}) < f\left(S^d_{dest}\right)$ then
      $S_{best} \leftarrow S_{const}$
    end if
  until runtime not expired
  return $S_{best}$
```

3.4. Initialization of the population

Memetic Algorithm has to start with a population of consistent chromosomes, i.e., each chromosome must be a feasible solution (see Definition 2). To create a diversified population, we used three algorithms to provide propitious individuals and a random generation for the rest of the population.

First algorithms are a greedy algorithm Gshifts and a daily global assignment algorithm AFshifts. These algorithms consist in constructing one solution by adding, step by step, an assignment (q,e) for Gshifts or a subset of assignments related to a day for AFshifts. We also use an improvement solution algorithm IDCshifts (on solutions provided by AFshifts or Gshifts) to provide chromosomes with good quality. It is based on an iterative Destruction/Construction algorithm which combines destruction and construction steps to explore the space of the solution neighborhood.

Gshifts. The principle of *Gshifts* is to construct a path of acceptable shifts for each employee on the horizon. For each employee $e \in E$ and for each day j of the horizon, we consider the set of non-assigned shifts of day j. Among these shifts, q is chosen such that $S \oplus (q,e)$ is feasible and $f(S \oplus (q,e))$ is minimum. As the arrangement of employees affects the structure of the solution, they are randomly considered. Insertion cost evaluation is done in $O(1)$. The complexity of Gshifts is $O(|E| \times |Q|)$.

AFshifts. It computes an optimal assignment (with minimum cost) for each day j, considering shifts of day j, $Q_{AVj} \subseteq Q$ and a set of available employees for this day j, $E_{AVj} \subseteq E$. So the global problem is decomposed day by day. Also in this case, the day order examination influences the structure of the solution, and so, they are randomly considered.

Let S_k be the current solution constructed after k iterations and consider we will construct the planning of day j. For each couple (q,e), $q \in Q_{AVj}$, $e \in E_{AVj}$, assignment cost $f(S_k \oplus (q,e))$ is computed in $O(1)$. $f(S_k \oplus (q,e)) = \infty$ if $S_k \oplus (q,e)$ is not a feasible solution. A classical resolution of assignment problem such as Hungarian algorithm (Kuhn, 1955) allows to compute a maximum matching, with minimum cost, between shifts and employees for the considered day. It provides a subset of feasible assignments $\mathcal{L} = \{(q_1, e_1), \ldots, (q_x, e_x)\}$. Because these assignments are feasible $S_k \bigoplus \mathcal{L}$ is a feasible solution (see Property 2). AFshifts gives a solution in $O(|J| \times |E|^3)$.

IDCshifts. Algorithm *IDCshifts* (see Algorithm 2) is based on the principle of the iterative Destruction/Construction algorithm proposed in Ruiz and Stützle (2007) and Bouly, Moukrim, Chanteur, and Simon (2008). As it is an improvement solution algorithm, it works on an existing solution $S = \langle A, Q_{AV}, E \rangle$. From S, it consists in first removing a subset of d assignments from S. These d removed assignments are randomly chosen. d is also randomly chosen between 1 and $\frac{|Q|}{|E|}$. For each destruction of assignment (q,e), the solution $S \ominus (q,e)$ respects all constraints (2)–(6) of Section 2.2. Let $\mathcal{L}$ be the set of the d assignments to remove from S. We denote $S^d_{dest} = S \bigominus \mathcal{L}$.

In a second step, the algorithm rebuilds solution, starting from S^d_{dest} by *Construction* process, that is an adaptation of the best insertion method (Ruiz & Stützle, 2007). This step consists in computing new assignments from available shifts (including available shifts provided by destruction step) and available employees (including those provided by destruction step). For each day j on the horizon, let Q_j be the set of non assigned shifts of day j and E_j the set of available employees of day j. For each day j and for each shift $q \in Q_j$, we determine the available employee $e \in E_j$ such as adding (q,e) provides best improvement of the solution quality. (q,e) is added to S^d_{dest}. Note that days, shifts and employees are randomly examined.

These two steps are alternatively iterated until a preset runtime is expired. The complexity of this algorithm is $O(|Q| \times |E|)$. IDCshifts will be used in our experimentations to evaluate the improvement of our Memetic Algorithm (i.e. $f(S_{IDCshifts})$ will be the reference value where $S_{IDCshifts}$ is the best solution found by IDCshifts).

Algorithm 3. Improvement Operator

```
Input:
  $C_0$: initial chromosome
Output:
  $C$: improved chromosome
  $S \leftarrow Dual(C_0)$
  repeat
    $ImpOP \leftarrow$ random choice in unmarked functions {REAS,
    COMP,DC}
    $S' \leftarrow ImpOP(S)$
    Mark the function corresponding to $ImpOP$
    if $f(S') < f(S)$ then
      $S \leftarrow S'$
      Unmark all functions
    end if
  until There is no unmarked function
  return $Dual(S)$
```

3.5. Local search as improvement operator

In GA, when a new chromosome is generated, mutation operator is used to alter some genes with a probability equal to mutation rate *pm*. The aim is to introduce some extra variability into the population. For *MA*, this mutation is processed by Local Search (LS) techniques whose purpose is the improvement of the considered solution. In our algorithm this Improvement Operator is a combination of three neighborhood functions dedicated to exploring the solution space differently. They are denoted *REAS*, *COMP* and *DC*. They are applied, alternatively and randomly, until there is no further improvement, as described in Algorithm 3. The shared principle of these functions is to compute new feasible assignments of shifts to employees, with or without prior destructions. The resulting solution is always better than or equivalent to the original one.

Reassignment operator REAS. It consists in randomly extracting for each employee e, an assignment $(q,e) \in A$ from the selected solution $S = \langle A, Q_{AV}, E\rangle$. We denote $S' = S \ominus (q,e)$. Then, for each of these extractions, *REAS* rebuilds a solution $S' \oplus (q',e)$ such that $q' \in Q_{AV} \cup \{q\}$, $S' \oplus (q',e)$ is feasible and $f(S' \oplus (q',e))$ is minimum. We note that, only employees with at least one assigned shift, are considered. The returned solution S_{REAS}, has a potential set of different assignments, but an equal number of non-assigned shifts and $f(S_{REAS}) \leqslant f(S)$ (see Example 3). The complexity of this process is $O(|Q| \times |E|)$.

Example 3. Let $S = \langle\{(q_1,b),(q_3,a),(q_4,b)\},\{q_2\},\{a,b,c\}\rangle$ be the new individual. We consider $S'_a = S \ominus (q_3,a)$ where (q_3,a) has been randomly chosen. Then operator *REAS* explore the following neighbor solution:

- $S'_a \oplus (q_2,a) = S_a = \langle\{(q_1,b),(q_2,a),(q_4,b)\},\{q_3\},\{a,b,c\}\rangle$ where S_a is feasible solution

Suppose $f(S_a) < f(S)$. So the process iterates for employee b with $f(S_a)$.

Note that c is not considered since there is no shift associated to it.

Complete operator COMP. It consists in attempting to add a new improving assignment (q',e), for each employee e such that there exists an assignment $(q',e) \in A$ of the selected solution $S = \langle A, Q_{AV}, E\rangle$ with $q' \in Q_{AV}$ and $S \oplus (q',e)$ is feasible. The returned solution S_{COMP}, has a potential increased set of assignments, and a smaller or equal number of non-assigned shifts and $f(S_{COMP}) \leqslant f(S)$ (see Example 4). The complexity of this process is $O(|Q| \times |E|)$.

Example 4. Let $S = \langle\{(q_1,b),(q_3,a),(q_4,b)\},\{q_2\},\{a,b,c\}\rangle$ be the new individual. We consider that $S \oplus (q_2,a)$ and $S \oplus (q_2,b)$ are feasible. Then operator *COMP* explores the following neighbor solutions:

- $\langle\{(q_1,b),(q_3,a),(q_4,b),(q_2,a)\},\emptyset,\{a,b,c\}\rangle$
- $\langle\{(q_1,b),(q_3,a),(q_4,b),(q_2,b)\},\emptyset,\{a,b,c\}\rangle$

Note that employee c is not considered with this operator.

Destruction Construction operator DC. The Destruction Construction operator is based on the principle of Iterative Destruction Construction algorithm previously presented in Algorithm 2. It consists in doing one iteration of IDCshifts. The complexity of this process is $O(|Q| \times |E|)$.

Example 5. Let $S = \langle\{(q_1,b),(q_2,a),(q_3,b),(q_4,c),(q_6,a)\},\{q_5,q_7,q_8\},\{a,b,c\}\rangle$ be the new individual. The number of removed assignment is $d = 2$. Then the two assignments (q_2,a) and (q_4,c) are randomly chosen and removed. We obtain the new solution $S_0 = \langle\{(q_1,b),(q_3,b),(q_6,a)\},\{q_2,q_4,q_5,q_7,q_8\},\{a,b,c\}\rangle$. Days of the horizon are randomly treated (order: $j3$, $j1$ and $j2$). The construction process explores the solution space as follows:

- Day $j3$: $Q_{j3} = \{q_7,q_8\}$ and $E_{j3} = \{a,b\}$.
  - DC compares $S \oplus (q_7,a)$ and $S \oplus (q_7,b)$ (both are feasible solutions), then retains $S \oplus (q_7,b)$
  - DC retains $S \oplus (q_8,a)$ because b is not yet available for day $j3$.
- Day $j1$: $Q_{j1} = \emptyset$ and $E_{j3} = \{a,c\}$.
- Day $j2$: $Q_{j2} = \{q_2,q_4,q_5\}$ and $E_{j2} = \{a,c\}$.
  - DC compares $S \oplus (q_5,a)$ and $S \oplus (q_5,c)$ (both are feasible solutions), then retains $S \oplus (q_5,a)$
  - DC retains $S \oplus (q_2,c)$ because a is not yet available for day j2.
  - No more available employee for shift q_4

We remind that days, shifts and employees are randomly examined. The resulting solution is:

$S = \langle\{(q_1,b),(q_2,c),(q_3,b),(q_5,a),(q_6,a),(q_7,b),(q_8,a)\},\{q_4\},\{a,b,c\}\rangle$

Algorithm 4. Memetic Algorithm (*MA*)

```
Input:
  $Q$: set of shifts
  $E$: set of employees
  $N$: population size
  $f$: fitness function
  $pm_0$: initial probability
  $\lambda$: constant
Output:
  $S_{best}$: best solution found

  $P \leftarrow Initialization(N)$ $\{P = \{C_0, C_1, \ldots, C_{N-1}\}\}$
  $P \leftarrow DescendingSort(P,f)$
  $pm \leftarrow pm_0$
  while runtime not expired do
    $C' \leftarrow Selection(P)$
    $C'' \leftarrow Selection(P \backslash \{C'\})$
    $C_{neo} \leftarrow Crossover(C', C'')$
    $prob \leftarrow$ random between 0 and 1
    if $prob \leqslant pm$ then
      $C_{neo} \leftarrow Improvement(C_{neo})$
    end if
    if $f(Dual(C_{neo})) \leqslant f(Dual(C_0))$ then
      if $\exists k \in [0, N-1]$ such that $f(Dual(C_{neo})) = f(Dual(C_k))$
      then
        $C_k \leftarrow C_{neo}$
        $pm \leftarrow \lambda \times pm$
      else
        Eject $C_0$ from $P$
        Insert $C_{neo}$ in $P$
        $pm \leftarrow pm_0$
      end if
    else
      $pm \leftarrow \lambda \times pm$
    end if
  end while
  $S_{best} \leftarrow Dual(C_{N-1})$
  return $S_{best}$
```

3.6. General algorithm

Our Memetic Algorithm is described by Algorithm 4. First, population P is initialized as mentioned in Section 3.4 with N

chromosomes $C_0, C_1, \ldots, C_{N-1}$. For more convenience, the population is sorted by descending fitness values f such that $\forall i \in [0, N-2], f(Dual(C_i)) \geqslant f(Dual(C_{i+1}))$.

At each iteration of the algorithm a couple of individuals, C' and C'', is chosen among the population using *Selection* operator (see Section 3.2). Then *Crossover* operator (see Section 3.3) is used to produce a child chromosome C_{neo}. C_{neo} could be improved by *Improvement* operator (see Section 3.5) with probability pm (initially preset to pm_0).

When C_{neo} ameliorates the population, then it integrates population. There is two possible cases for this configuration: either there is a chromosome C_k with a same fitness, then C_{neo} replaces C_k, or C_{neo} strictly improves the population, then the worst chromosome C_0 is removed and C_{neo} is inserted.

Probability of child improvement, pm, evolves following the *fitness* of the new generated individuals. While there is no strict improvement pm decreases ($pm = \lambda \times pm, 0 < \lambda < 1$), but once the population is improved pm is reset to pm_0. To sum up, population P is changed step by step, and the process is iterated for a fixed running time. Algorithm *MA* returns solution S_{best} corresponding to the best chromosome of the population, C_{N-1}, at the end of the process.

4. Numerical results

All algorithms have been implemented in java and tested on Intel Xeon Quad Core at 2.4 Ghz. They have been trained on instances which are extracted from real cases.

The main characteristics of our 13 instances are given in Table 1. Columns are respectively, *Num* (instance number), *E* (number of employees), *C* (number of skills), *Q* (number of shifts), *QMax* (maximum number of shifts for a day), *ContH* (number of types of employment contracts), $\frac{|Q|}{|E|}$ (average number of shifts per employee) and $\bar{Q}_{day}$ (average number of shifts per day). The rostering horizon considered for instances is a month (from 28 to 31 days). Instances cannot be more detailed due to confidential reasons.

4.1. Quality of the initial population: Gshifts, AFshifts & IDCshifts results

We have experimented two greedy algorithms (Gshifts, AFshifts) and a dedicated Local Search (IDCshifts) to provide good initial individuals (see Section 3.4).

First, we compare results obtained by algorithms Gshifts and AFshifts. For each instance *Num*, we present in Table 2, the improvement of AFshifts relatively to Gshifts, $\Delta^{Gshifts}_{AFshifts}$ computed as described in Eq. (11). This equation quantifies the improvement of Algorithm 2 compared to Algorithm 1.

Table 1
Instance characteristics.

Num	*E*	*C*	*Q*	*QMax*	*ContH*	$\frac{\lvert Q\rvert}{\lvert E\rvert}$	$\bar{Q}_{day}$
1	277	6	2343	141	14	8.46	75.58
2	197	6	1998	95	2	10.14	64.45
3	297	6	2425	153	14	8.16	78.23
4	224	5	2206	109	1	9.85	71.16
5	323	5	2065	109	1	6.39	66.61
6	289	3	1625	92	3	5.62	42.42
7	148	5	1089	57	2	7.36	35.13
8	32	3	496	16	2	15.50	16.00
9	294	6	3449	197	1	11.73	111.26
10	104	5	1027	55	2	9.88	33.13
11	48	5	713	23	3	14.85	23.00
12	241	6	3777	122	14	15.67	121.84
13	227	6	3818	125	6	16.82	123.16

Table 2
Improvement of AFshifts in comparison of Gshifts, computation time of Gshifts, computation time of AFshifts and improvement of IDCshifts in comparison of AFshifts.

Num	$\Delta^{Gshifts}_{AFshifts}$ (%)	$T_{Gshifts}$ (s)	$T_{AFshifts}$ (s)	$\Delta^{AFshifts}_{IDCshifts}$ (%)
1	30.79	33	78	4.78
2	42.57	28	47	7.73
3	36.74	36	80	5.61
4	38.60	32	63	2.49
5	19.81	29	74	0.78
6	19.30	17	49	1.16
7	19.10	9	21	5.06
8	35.59	6	8	40.38
9	34.59	44	127	15.43
10	20.15	9	19	17.12
11	29.31	7	9	40.42
12	40.59	63	149	34.04
13	40.85	58	151	26.00

$$\Delta^{Algorithm1}_{Algorithm2} = \frac{f(S_{Algorithm1}) - f(S_{Algorithm2})}{f(S_{Algorithm1})}$$

where $S_{Algorithm}$ is the best solution found by *Algorithm*

AFshifts provides a solution in a single minute in average ($T_{AFshifts}$) whereas Gshifts is executed on less than one minute ($T_{Gshifts}$) in average, in accordance to their complexity, but AFshifts provides each time a better solution than Gshifts one.

Our third heuristic IDCshifts requires acceptable solution. We use AFshifts for each experiment in order to obtain this initial solution since it gives the best results in the first step.

To assess the improvement, IDCshifts has been executed for each instance 10 times during 5 min. We retained the best solution for each instance and compared it with AFshifts one. Column $\Delta^{AFshifts}_{IDCshifts}$ in Table 2 reports values of these comparisons. On average, quality improvement is 15.46%.

4.2. MA experimentations

To set the most appropriate population size, we have tested *MA* for different values, and compared the improvement gap from a basic solution supplied by IDCshifts.

We denote *MA*(x) the Memetic Algorithm when the population size is set to x. In Table 3, we report $\Delta^{IDCshifts}_{MA(x)}$ for sizes $x \in \{5, 10, 15, 20, 25\}$. Populations are initialized as described in Section 3.4. For each instance, *MA* was performed 10 times during 30 min for each run. Sub-column *max* is related to the best improvement among all executions, and *avg* is related to the average improvement. The mutation rate is $pm_0 = 0.75$ and its decreasing ratio is $\lambda = 0.99$. The D_{max} value of operator DC is fixed to $\frac{|Q|}{|E|}$.

It appears that *MA* did not improve the results for the instances 1 to 6 and 9. Indeed, IDCshifts provides good results for theses instances, that can be explained by the large difference between $\bar{Q}_{day}$ and *QMax*. The greater is this difference, the less the workload is uniform on the horizon. Indeed, the main IDCshifts feature is a random exploration of solution space. Moreover as $\frac{|Q|}{|E|}$ is small for these instances the assignment optimization is easier.

Nevertheless the improvement is significant for instances 8, 11–13 (more than 50% for instances 8 and 12). The gap between $\bar{Q}_{day}$ and *QMax* (see Table 1) is close to zero and then reveals a uniform workload structure. So for a given available shift there are many eligible employees that allow to explore a larger neighborhood for *REAS* and *COMP* functions. Moreover both are descent methods so improve the solution.

For instances 7 and 10, the improvement is weak as the gap between $\bar{Q}_{day}$ and *QMax* is close to easier instances. In the following, only significant instances (7, 8, 10–13) will be considered.

Table 3
MA best improvement and improvement on average compared with IDCshifts solution w.r.t. population size (best results in bold).

Num	$\Delta^{IDCshifts}_{MA(5)}$ (%)		$\Delta^{IDCshifts}_{MA(10)}$ (%)		$\Delta^{IDCshifts}_{MA(15)}$ (%)		$\Delta^{IDCshifts}_{MA(20)}$ (%)		$\Delta^{IDCshifts}_{MA(25)}$ (%)	
	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
7	0.87	0.73	0.85	0.63	0.97	**0.76**	0.78	0.62	**0.96**	0.75
8	51.65	49.76	49.55	47.63	50.59	48.64	**51.94**	**50.26**	51.81	49.40
9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
10	**4.14**	**3.55**	3.33	2.85	3.30	2.98	3.94	3.00	3.18	2.50
11	32.08	30.91	33.95	32.60	35.45	32.30	**35.69**	**33.30**	34.16	32.40
12	**52.76**	49.52	51.92	50.06	50.46	48.84	51.58	**50.26**	50.29	48.84
13	43.31	**41.46**	42.60	40.94	42.52	41.05	**43.99**	41.09	41.48	39.20

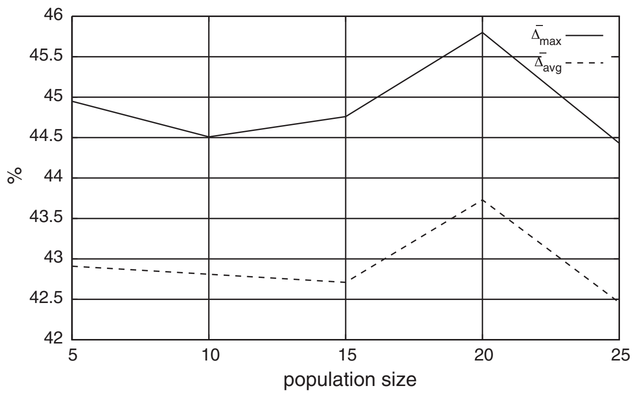

Fig. 3. Improvement over the size of the population.

In Fig. 3, $\bar{\Delta}_{\max(x)}$ (respectively $\bar{\Delta}_{avg(x)}$) represents the average of the best improvements for the instances (respectively the same for the improvement averages) for population of size *x*. $\bar{\Delta}_{\max(x)}$ and $\bar{\Delta}_{avg(x)}$ are computed from Table 3. It brings out that a population of 20 individuals is best suited for instances 8, 11 and 12, on average. For other instances, the gap between the best improvement and improvement obtained for a population size of 20 individuals is tiny. So, choose a population size of 20 individuals as reference is convenient for our staff scheduling problem.

5. Conclusion

In this paper, we have dealt with a staff scheduling problem in airport security service, with a lot of specificities. We defined, closely with planning operators, an economic function that allows to evaluate plannings with relevance. Also, we have proposed a Memetic Algorithm *MA* to solve this problem. We first have modeled the problem and give a formal definition of the solutions. We proposed dedicated heuristics Gshifts, AFshifts and IDCshifts to generate the initial population and a basic tournament selection operator to pick the parents in the reproduction process. We have presented a devoted crossover operator and a suitable improvement operator, made of three neighborhood functions: *REAS*, *COMP* and *DC*.

Our *MA* algorithm proved its efficiency by the quality of provided plannings. *MA* always provides better solutions than those produced by the current software. Let us note that the planning operators manually adapt solutions supplied by this one. Perspectives related to this work are various. Among them, other elementary operations on planning could be defined but also others neighborhoods could be included for our Memetic Algorithm.

Acknowledgments

We thank Mr. Vathana TIV, Senior Director Operational Efficiency of ICTS Europe Holdings BV, for his contribution in this project.

References

Abdoul Soukour, A., Devendeville, L., & Lucet, C. A. M. (2012). Staff scheduling in airport security service. In *14th IFAC symposium on information control problems in manufacturing (INCOM'12)*.

Ásgeirsson, E. I. (2012). Bridging the gap between self schedules and feasible schedules in staff scheduling. *Annals of Operations Research*, 1–19. http://dx.doi.org/10.1007/s10479-012-1060-2.

Aykin, T. (1998). A composite branch and cut algorithm for optimal shift scheduling with multiple breaks and break windows. *JORS, 49*(6), 603–615.

Bechtold, S. E., & Jacobs, L. W. (1990). Implicit modeling of flexible break assignments in optimal shift scheduling. *Management Science, 36*(11), 1339–1351.

Blöchliger, I. (2004). Modeling staff scheduling problems. A tutorial. *European Journal of Operational Research, 158*, 533–542.

Bouly, H., Dang, D.-C., & Moukrim, A. (2010). A memetic algorithm for the team orienteering problem. *4OR, 8*(1), 49–70.

Bouly, H., Moukrim, A., Chanteur, D., & Simon, L. (2008). Un algorithme de destruction/construction itératif pour un problème de tournées de véhicules spécifique. In *Conférence internationale de modélisation, optimisation et simulation*.

Brucker, P., Qu, R., & Burke, E. (2011). Personnel scheduling: Models and complexity. *European Journal of Operational Research, 210*, 467–473.

Detienne, B., Péridy, L., Pinson, E., & Rivreau, D. (2009). Cut generation for an employee timetabling problem. *European Journal of Operational Research, 197*, 1178–1184.

Ernst, A. T., Jiang, H., Krishamoorthy, M., & Sier, D. (2004). Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research, 153*, 3–27.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness. Series of books in the mathematical sciences*. W.H. Freeman.

Genç, H. M., Erol, O. K., Eksin, I., Berber, M. F., & Güleryüz, B. O. (2012). A stochastic neighborhood search approach for airport gate assignment problem. *Expert Systems with Applications, 39*, 316–327.

Günther, M., & Nissen, V. (2012). Application of particle swarm optimization to the british telecom workforce scheduling problem. In *Proceedings of the ninth international conference on the practice and theory of automated timetabling (PATAT)* (pp. 242–255).

Holland, J. H. (1992). *An introductory analysis with applications to biology, control and artificial intelligence*. MIT Press (chap. Adaptation in natural and artificial system, pp. 219–234).

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistic Quarterly, 2*, 83–97.

Moscato, P. (1999). *New ideas in optimization*. McGraw-Hill (chap. Memetic Algorithms: A short introduction, pp. 219–234).

Oliver, I. M., Smith, D. J., & Holland, J. R. C. (1987). A study of permutation crossover operators on the traveling salesman problem. In *Second international conference on genetic algorithms and their application* (pp. 224–230).

Portmann, M. -C., & Vignier, A. (2000). Performances' study on crossover operators keeping good schemata for some scheduling problems. In *Genetic and evolutionary computation conference* (pp. 331–338).

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & OR, 31*(12), 1985–2002.

Rekik, M., Cordeau, J.-F., & Soumis, F. (2004). Using benders decomposition to implicitly tour scheduling. *Annals of Operations Research, 128*, 111–133.

Ruiz, R., & Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research, 177*, 2033–2049.

Soumis, F., Pesant, G., & Rousseau, L.-M. (2005). *Gestion de production et ressources humaines*. Presses Internationales Polytechnique (chap. 4. Gestion des horaires et affectation du personnel, pp. 71–110).