Available online at www.sciencedirect.com

ScienceDirect

European Journal of Operational Research 195 (2009) 75–88

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

Discrete Optimization

The dynamic frequency assignment problem

Audrey Dupont [a], Andréa Carneiro Linhares [a], Christian Artigues [b,*], Dominique Feillet [a], Philippe Michelon [a], Michel Vasquez [c]

[a] Laboratoire d'Informatique d'Avignon, Université d'Avignon, Agroparc BP 1228, 84911 Avignon Cedex 9, France
[b] Université de Toulouse, LAAS CNRS, 7 Avenue du Colonel Roche, 31077 Toulouse Cedex 4, France
[c] Centre de Recherche LGI2P, Ecole des Mines d'Alès, Site EERIE, Parc Scientifique Georges Besse, 30035 Nîmes Cedex 01, France

Received 1 December 2006; accepted 22 January 2008
Available online 1 February 2008

Abstract

In this paper, we consider a frequency assignment problem occurring in a military context. The main originality of the problem pertains to its dynamic dimension: new communications requiring frequency assignments need to be established throughout a battlefield deployment. The problem resolution framework decomposes into three phases: assignment of an initial kernel of communications, dynamic assignment of new communication links and a repair process when no assignment is possible. Different solution methods are proposed and extensive computational experiments are carried out on realistic instances.
© 2008 Elsevier B.V. All rights reserved.

Keywords: Frequency assignment; Dynamic problem; Heuristics; Tabu search and consistent neighborhood; Branch&Bound

1. Introduction

Using efficiently the radio spectrum has become of critical importance with the development of new communication technologies as mobile telephony, radio and TV broadcasting, or satellite. This issue has raised a growing interest in the literature over the years, giving rise to the so-called Frequency Assignment Problems. In these problems, frequencies have to be assigned to Hertzian communications, such that interferences between communications are avoided. Many models exist depending on the characteristics of the Hertzian network, the way electromagnetic interferences are modeled or the objectives pursued.

In this paper, we consider a frequency assignment problem occurring in a military context. This problem was submitted by the CELAR (Centre ELectronique de L'ARmement) and concerns the assignment of frequencies to Hertzian communications in the course of a military deployment. The main originality of the problem pertains to its dynamic dimension. New communications need to be established throughout the deployment and require new frequency assignments.

The communication network progresses as follows. An initial kernel is first installed; antennas are laid on the battleground and links are established between some pairs of antennas to permit communications. New communication links are then progressively established, between existing or new antennas.

This deployment scheme gives rise to three types of frequency assignment problems. The first one corresponds to the assignment of frequencies to the communications constituting the initial kernel. This is a standard static frequency assignment problem. The second one relates to the dynamic assignment of frequencies to the new communication links. Though

* Corresponding author. Tel.: +33 5 61 33 79 07; fax: +33 6 61 33 69 36.
E-mail addresses: Audrey.Dupont@univ-avignon.fr (A. Dupont), Andrea.Linhares@univ-avignon.fr (A.C. Linhares), artigues@laas.fr (C. Artigues), Dominique.Feillet@univ-avignon.fr (D. Feillet), Philippe.Michelon@univ-avignon.fr (P. Michelon), Michel.Vasquez@ema.fr (M. Vasquez).

0377-2217/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2008.01.028

changing frequencies assigned previously is technically possible, it is quite costly in terms of time and human resources. The strategy proposed by the CELAR was to avoid changing frequencies, unless no other solution exists. Hence, the second problem consists in assigning frequencies to new links avoiding interferences until no frequency can be assigned to a link. We call this last situation a blocking. In the case of a blocking, a third problem occurs. One has to reassign frequencies so that a consistent assignment is recovered. Several objectives can be pursued. The retained objective consists in minimizing the number of reallocated frequencies.

We call the succession of frequency assignment problems encountered here the Dynamic Frequency Assignment Problem (*DFAP*). Though frequency assignment problems have been largely explored in the literature (see, e.g., Aardal et al., 2003, 2007), the dynamic and the repair features met here make the *DFAP* a new problem.

After presenting more precisely the context of our study in Section 2, we give a detailed description of the problem in Section 3. The different methods used to solve the three problem components are then described in Section 4. Finally, Section 5 evaluates these methods on a set of scenarios provided by the CELAR.

2. Context and related work

The study of Hertzian frequency assignment problems in a military context recently gave a new rise to this field of research. This paper on the *DFAP* follows a large amount of work devoted to these types of problems in this context.

The first one, and probably the most famous, is issued from the European project CALMA (*Combinatorial ALgorithms for Military Applications*). During this project, many concepts from Computing Science, Mathematics of Operations Research and Local Search were applied to the Radio Link Frequency Assignment Problem. The purpose of the project was to define and use a standard problem as a tested for the development and comparison of various optimization strategies and methods. Results were compiled in several survey papers (Aardal et al., 2002, 2003, 2007).

The CELAR then proposed to study several more realistic modelings. A first one includes antenna orientations and the possibility to progressively relax interference constraints. This new model was the subject of the ROADEF 2001 challenge; a detailed description of this challenge can be found on the dedicated web site.[1] Several approaches were developed and permitted to address the new modeling issues quite efficiently (Dupont et al., 2004; Hertz et al., 2005).

Another proposed model includes a more precise consideration of interference phenomena by introducing a global constraint simulating the influence of all the transmitters which potentially disturb a receiver (Palpant et al., 2002, 2008).

With the *DFAP*, the CELAR returns to a standard modeling of the problem, but introduces the above mentioned dynamic dimension. Solution algorithms then have to deal with unknown (future) data. The literature about uncertain data presents two extreme cases:

the data is entirely accessible but imprecisely;
the data is partially accessible.

The first case pertains to the field of robust optimization (Kouvelis and Yu, 1997). The issue is then to propose solutions able to absorb adequately different realizations of data. The second case, encountered here, corresponds to dynamic or "on-line" optimization (Halldorsson and Szegedy, 1992). It has the three following characteristics:

new decisions to be taken arise one by one;
decisions are irrevocable;
no knowledge of the future is accessible.

In these two cases of uncertainty, a situation where the current solution becomes unacceptable can arise. It is then necessary to consider the reappraisal of the previous decisions (for example, those taken by the on-line algorithm). The implemented techniques are called solution repair techniques. They are aimed at recovering the solution admissibility, generally by minimizing the number of decision reappraisals.

Dynamic problems are widely studied within the framework of various optimization problems, such as Bin Packing (Coffman et al., 1983; Grove, 1995), Scheduling (Elkhyari et al., 2004), Graph Coloring (Vishwanathan, 1990) or Vehicle Routing Problems (Psaraftis, 1988; Gendreau et al., 1999). In the context of Frequency Assignment Problems, several works are also devoted to on-line algorithms (Fotakis et al., 1999; Crescenzi et al., 2000; Daniels et al., 2004; Fitzpatrick et al., 2004). However, these works are focused on cellular networks, which define quite different problems than the one considered in our study.

[1] http://www.prism.uvsq.fr/vdc/ROADEF/CHALLENGES/2001/challenge2001_en.html.

3. Problem description

3.1. Formal description

A Hertzian Communication network is composed of a set of radio links connecting different strategic sites. A link between two antennas is divided into two paths, in both communication directions. Hence, a path is a vector defined by its emitter site and its receiver one. Establishing a radio link requires the assignment of a frequency to each of its paths.

A formal description requires the introduction of preliminary notations. We call:

S, the set of the geographical sites of the network;
n, the total number of links (obtained at the end of the deployment);
$L = \{l_0, \ldots, l_{n-1}\}$, the set of links;
$P = \{p_0, \ldots, p_{2n-1}\}$, the set of paths; link l_i is composed of paths p_{2i} and p_{2i+1}.

All the paths of P have the same frequency domain D, but two different domains are investigated. The first one D_1 is made up of six intervals: $D_1 = [40,000, 40,140] \cup [41,000, 41,140] \cup [42,000, 42,140] \cup [43,000, 43,140] \cup [44,000, 44,210] \cup [45,000, 45,210]$; the frequency gap between two channels is equal to 70. D_1 thus contains 20 frequency values (3 in the first four intervals and 4 in the two last). The second domain D_2 is composed of a unique interval $D_2 = [40,000, 41,890]$, also with a frequency gap of 70. D_2 contains 28 frequency values (see Fig. 1). In the following, we note D the domain, which can be either D_1 or D_2.

In order to ensure a good quality of communication in the radio network, interferences between links must be avoided. Interference constraints are defined as follows. When two paths $p_i, p_j \in P$ are close enough to cause interferences, a binary constraint C_{ij} enforcing a gap between p_i and p_j is defined: $|f_i - f_j| \geqslant g_{ij}$, where f_i is the frequency of the path p_i, f_j the frequency of p_j and g_{ij} the required gap.

The value g_{ij} depends on the relation between p_i and p_j:

the *duplex* constraints between two paths belonging to a same link impose a gap $g_{ij} = 600$;
the *co-site* constraints between two paths connected to a same site contain:
  the *transmitter–receiver* constraints, where the gap is $g_{ij} = 220$;
  the *transmitter–transmitter* constraints, where the gap is $g_{ij} = 100$;
  the *receiver–receiver* constraints, where the gap g_{ij} belongs to [17, 103] for our instances with a mean value greater than 70;
the far-field constraints, where the gap g_{ij} practically never exceeds 70.

Note that duplex and transmitter–receiver co-site constraints require to assign frequencies to different intervals when the domain considered is $D = D_1$.

Fig. 2 illustrates the notion of sites, antennas, communication links, as well as the different types of constraints.

Due to material specificities, the maximal number of links on a site is 8. Such a site, called *Cart 8*, involves 16 paths and a lot of difficult constraints (duplex and co-site). Very few possibilities exist to avoid interferences. Assigning frequencies is then often a difficult task, especially in an on-line setting, when it is not known in advance that the site is going to be a Cart 8.

The *DFAP* begins with an initial set of links, called the initial kernel. The first subproblem is to assign frequencies to the paths constituting the kernel. Then, new links arrive dynamically. Once a link arrives, frequencies have to be assigned, in an on-line fashion. This defines the second subproblem. In case of blocking, i.e. when no allocation is possible, a repair procedure must reallocate some paths. A first objective is to minimize the number of times the repair procedure is called. Then, the number of repaired paths must be minimized.

During this process, the desired computing time is limited to only a few seconds for assigning every new link, and no more than a few minutes for each call to the repair procedure. These delays correspond to acceptable waiting times for users of the system during a military deployment.

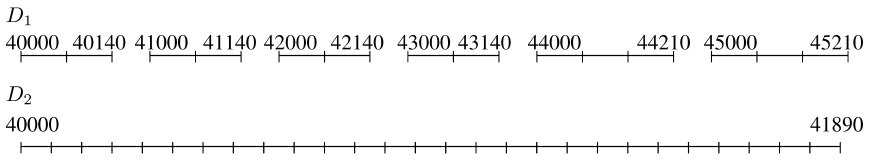

Fig. 1. Domain characteristics.

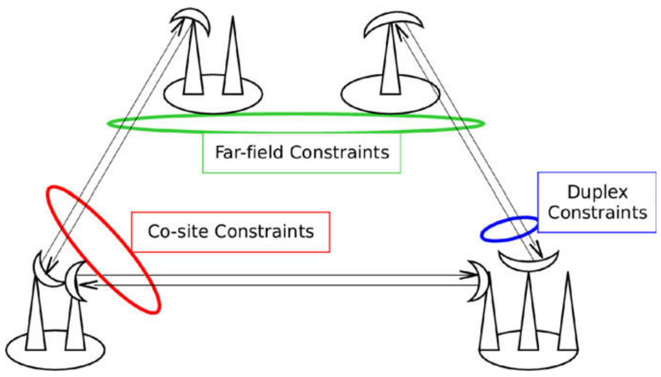

Fig. 2. Example of a deployment.

3.2. Modeling

The three underlying subproblems of the *DFAP* can be described as follows.

3.2.1. Static model

This model corresponds to frequency assignment for the initial kernel:

$$\begin{aligned}&\text{find}\quad f_0,\ldots,f_{2k-1}\quad\text{so that:}\\ &|f_i-f_j|\geqslant g_{i,j}\quad (p_i,p_j\in\{p_0,\ldots,p_{2k-1}\}),\\ &f_i\in D\quad (p_i\in\{p_0,\ldots,p_{2k-1}\}),\end{aligned}$$

where k is the number of links in the initial kernel.

3.2.2. Dynamic model

This second model corresponds to the on-line situation. When a new link $l_i=(p_{2i},p_{2i+1})\in L$ needs to be established, the problem to solve is:

$$\begin{aligned}&\text{find}\quad f_{2i},f_{2i+1}\ \text{so that:}\\ &|f_{2i}-f_j|\geqslant g_{2i,j}\quad (p_j\in\{p_0,\ldots,p_{2i-1}\}),\\ &|f_{2i+1}-f_j|\geqslant g_{2i+1,j}\quad (p_j\in\{p_0,\ldots,p_{2i-1}\}),\\ &|f_{2i+1}-f_{2i}|\geqslant g_{2i+1,2i},\\ &f_{2i},f_{2i+1}\in D,\end{aligned}$$

where $f_0,\ldots,f_{2i-1}$ have been assigned during the previous iterations and during the initial kernel assignment.

3.2.3. Repair model

This last model corresponds to the repair process. When a blocking occurs for a new link $l_i=(p_{2i},p_{2i+1})\in L$, the problem to optimize is

$$\begin{aligned}\text{Min}\quad & (|\{h\in\{0,\ldots,2i-1\}\setminus f'_h\neq f_h\}|),\\ \text{s.t.}\quad & |f_{2i}-f'_j|\geqslant g_{2i,j}\quad (p_j\in\{p_0,\ldots,p_{2i-1}\}),\\ & |f_{2i+1}-f'_j|\geqslant g_{2i+1,j}\quad (p_j\in\{p_0,\ldots,p_{2i-1}\}),\\ & |f'_h-f'_j|\geqslant g_{hj}\quad (p_h,p_j\in\{p_0,\ldots,p_{2i-1}\}),\\ & |f_{2i+1}-f_{2i}|\geqslant g_{2i+1,2i},\\ & f_{2i},f_{2i+1}\in D,\\ & f'_h\in D\quad (p_h\in\{p_0,\ldots,p_{2i-1}\}),\end{aligned}$$

where $f_0,\ldots,f_{2i-1}$ have been assigned during the previous steps. Note that this repair problem includes the new link assignment.

4. Solving methodologies

The splitting of the *DFAP* into three subproblems implies to develop three different solution procedures. These approaches will be described in the subsequent subsections. However, the hybrid metaheuristic $\mathscr{CN} - Tabu$ is used to solve two of the three subproblems. For a better understanding of the methods, we first present $\mathscr{CN} - Tabu$.

4.1. $\mathscr{CN}$-Tabu

$\mathscr{CN} - Tabu$, standing for Consistent Neighborhood in a Tabu Search, was presented in Vasquez et al. (2005) for the solution of Constraint Satisfaction Problems. It is an original hybrid Tabu Search algorithm dealing with partial assignments of paths. A key point of the method is that the set of partial assignments considered is restricted to the assignments satisfying the set of constraints. The objective is then to find a complete solution. A partial configuration s is denoted $s = ((i_1, f_{i_1}), (i_2, f_{i_2}), \ldots, (i_{n_s}, f_{i_{n_s}}))$, with $0 \leqslant i_1 < i_2 < \cdots < i_{n_s} \leqslant 2n - 1$ and $0 \leqslant n_s \leqslant 2n - 1$. The cost function used is merely the number of assigned paths, i.e., the value of n_s.

Each move consists of two steps. First, a path with no currently assigned frequency is chosen, and a frequency is assigned to this path. By propagation on the constraint set, this new assignment may create conflicts with other paths. The consistency of the configuration is then restored, by simply removing the conflicting frequencies.

The neighborhood is evaluated efficiently using the incremental computing principle proposed in Fleurent and Ferland (1996). A tabu list is introduced to prevent cycling, which is especially likely to occur when n_s is close to $2n - 1$. This tabu list forbids temporarily every frequency of the neighbor paths that would cause a conflict with the new assignment (although these frequencies are not currently assigned). An aspiration criteria allows however the forbidden frequencies when the assignment will improve the cost function (i.e., increases n_s).

The starting configuration is obtained using the simple greedy heuristic presented in Section 4.3.1.

4.2. Kernel assignment

The kernel assignment problem is a small Constraint Satisfaction Problem. In our instances, the initial network is composed of 17 links. We solve it using the $\mathscr{CN} - Tabu$ approach, which proved to be able to find a feasible solution quickly for all our instances. The obtained kernel assignments are called *Cnt kernels* hereafter.

Obviously, since this initial assignment is computed without taking into account the links to appear later on, it may be that this initial solution leads to a poor quality assignment on the overall problem. In order to evaluate the influence of the initial assignment, we also solve the whole network as a static problem with $\mathscr{CN} - Tabu$ and extract the kernel. Such an approach, practically infeasible, provides reference kernel assignments called *Sol kernels*. These kernels are optimized in the sense that they can be extended to a solution of the entire problem, which is not necessarily the case of *Cnt kernels*.

4.3. On-line assignment

For the on-line assignment phase, the context imposes to use algorithms fixing the decisions in a sequential way, and forbidding any modification of the previous decisions. Such methods are generally called *greedy algorithms*. In an on-line situation, the order in which the decisions are treated is also imposed. Here, one has to assign frequencies to the two paths constituting a new link. So, the only strategy to define is the heuristic used to choose these frequencies.

The greedy algorithm treats iteratively new assignment requests (establishment of new links), as long as an assignment is possible. Then, it hands over to the repair method, which tries to restore the consistency of the network increased by the new link. As soon as the solution is repaired (with the new link assigned), the next links are considered, using once more the greedy algorithm.

4.3.1. Classical greedy

Many strategies are proposed in the literature for frequency assignment problems. Essentially, they define various criteria to decide which path to consider next. Then, two basic strategies are used to assign a frequency to these paths: select the minimal consistent frequency or select the most occupied one. The aim of these strategies is in both cases to maximize the remaining space in the domain. These two strategies can be declined in several ways (select the maximal frequency, any occupied frequency, the most occupied interval, combine these objectives lexicographically, etc.).

In this work, we evaluated about 10 variants of these two strategies. It turns out that the best one here is simply to select the minimal consistent frequency. In our context, this heuristic is slightly complicated, seeing that each iteration of the on-line algorithm necessitates to assign two frequencies. It is implemented as follows: at each iteration, the algorithm selects the consistent pair of frequencies with the minimal highest value. In case the highest values are equal, the pair with the minimal smallest value is chosen. Algorithms 1 and 2 detail a simple implementation of this heuristic.

```
Algorithm 1: Greedy Min-Freq($p_{2i}, p_{2i+1}$)
begin
    $(f_1^*, f_2^*) \leftarrow (+\infty, +\infty)$;
    for each pair $(f_1, f_2) \in D \times D$ do
        if $(f_1, f_2)$ is consistent then
            $(f_1^*, f_2^*) \leftarrow$ Min-Freq($(f_1, f_2), (f_1^*, f_2^*)$);
    if $(f_1^*, f_2^*) \neq (+\infty, +\infty)$ then
        $(f_{2i}, f_{2i+1}) \leftarrow (f_1^*, f_2^*)$;
end
```

```
Algorithm 2: Min-Freq($(f_1, f_2), (f_1^*, f_2^*)$)
begin
    if $(\max(f_1, f_2) < \max(f_1^*, f_2^*))$ then
        return $(f_1, f_2)$
    if $(\max(f_1, f_2) = \max(f_1^*, f_2^*))$ and $(\min(f_1, f_2) < \min(f_1^*, f_2^*))$ then
        return $(f_1, f_2)$
    return $(f_1^*, f_2^*)$
end
```

In Algorithm 1, all the consistent pairs of frequencies for the new link are compared with the *Min-Freq* criterion. This criterion is detailed in Algorithm 2. Two consistent frequency pairs are compared, and the best one is returned.

4.3.2. Availability greedy

The second implemented greedy algorithm is based on an original criterion that we call *site availability*.

Roughly, site availability is a measure of the potential for connecting new links to a site. When a new link is considered, the algorithm aims at choosing frequencies so that the sites connected to this link maintain a maximum level of availability. The motivation for using this measure relies on the structure of the interference constraints, which essentially imply co-site links; hence, the assignment of frequencies to a new link mainly depends on the frequencies assigned to other links connected to the same sites. Contrary to classical greedy heuristics, the problem is not handled through an aggregated measurement of the occupation of the domain, but rather considers the domain for each site separately. Presenting the algorithm requires to introduce some definitions.

The *availability* in emission $avail_E(s,f) \in \{0,1\}$ of a site $s \in S$ for a frequency f indicates whether frequency f can be assigned to a new path in emission from s or not. Actually, this value can only be computed approximately. Indeed, interference constraints to be taken into account should be: transmitter–receiver, transmitter–transmitter and far-field constraints. The two first ones all imply a fixed gap, namely 220 and 100, respectively, but far-field constraints depend on the geographic positioning of the link. In the computation of $avail_E(s,f)$, we neglect these constraints. This assumption makes sense since far-field constraints have far less impact than other types of constraints.

With this assumption, computing $avail_E(s,f)$ is easy. One only has to check whether any path in emission from s is assigned to a frequency belonging to $]f-100, f+100[$ and any path in reception is assigned to a frequency belonging to $]f-220, f+220[$.

Equivalently, the *availability* in reception $avail_R(s,f) \in \{0,1\}$ of a site $s \in S$ for a frequency f indicates whether the frequency f can be assigned to a new path in reception on s or not. Again, far-field constraints are neglected. Also, receiver–receiver constraints are assumed to impose a gap g with $70 < g < 140$, which is true most of the times and which permits to compute $avail_R(s,f)$ seeing that the step between successive values of D is 70.

The *availability* in emission (resp. reception) of a site $s \in S$, written $avail_E(s)$ (resp. $avail_R(s)$), is the number of frequencies $f \in D$ available in emission (resp. reception):

$$avail_E(s) = \sum_{f \in D} avail_E(s,f),$$

$$avail_R(s) = \sum_{f \in D} avail_R(s,f).$$

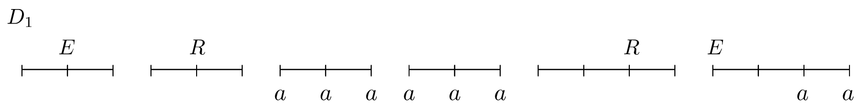

Fig. 3. Example of a site s with $avail_E(s) = 8$.

Finally, the availability $avail(s)$ of a site $s \in S$ is the sum of the availabilities in emission and reception:

$$avail(s) = avail_E(s) + avail_R(s).$$

Note that this measure gives an approximation of the network capability in receiving new links in the future, but that its value does not indicate the number of new links that could be added.

Fig. 3 illustrates the notion of site availability, in the case of a domain of the D_1 type. Frequencies used by paths previously assigned are mentioned with letters E and R respectively, for paths in emission and in reception. Frequencies f available in emission (i.e., such that $avail_E(s,f) = 1$) are indicated with a a.

The greedy heuristic is defined as the classical greedy heuristic of Subsection 4.3.1, replacing the minimal frequency criterion with the site availability measure. In case of equivalent availability, the *Min-Freq* criterion is used. The measure *Avail* is described in Algorithm 3. The complete heuristic is obtained from Algorithm 1 by changing the *Min-Freq* criterion to *Avail*.

```
Algorithm 3: Avail($(f_1, f_2)$, $(f_1^*, f_2^*)$)
begin
    if $(avail(f_1, f_2, s_1) + avail(f_1, f_2, s_2) > avail(f_1^*, f_2^*, s_1) + avail(f_1^*, f_2^*, s_2))$ then
        return $(f_1, f_2)$
    if $(avail(f_1, f_2, s_1) + avail(f_1, f_2, s_2) = avail(f_1^*, f_2^*, s_1) + avail(f_1^*, f_2^*, s_2))$ then
        return Min-Freq($(f_1, f_2)$, $(f_1^*, f_2^*)$)
    return Min-Freq($(f_1, f_2)$, $(f_1^*, f_2^*)$)
end
```

Here, $avail(f\ ,f\ ,s)$ represents the availability of s, when links l to l are assigned

Here, $avail(f_1,f_2,s)$ represents the availability of s, when links l_0 to l_{i-1} are assigned and link $l_i = (p_{2i}, p_{2i+1})$ is assigned to (f_1,f_2); the two sites incident to l_i are noted s_1 and s_2. Note that the consistency of (f_1,f_2) is evaluated on the real values of the constraints (and includes all types of constraints) since these values are known for the new link.

4.4. Repairs

Repair is used when a new link $l_i = (p_{2i}, p_{2i+1})$ cannot be allocated. The objective is then to re-assign a minimal number of paths in $\{p_0, \ldots, p_{2i-1}\}$. In the following, s_{cur} denotes the current configuration. Note that in this configuration paths p_{2i} and p_{2i+1} are not allocated.

We address the repair process with several approaches.

4.4.1. Repair by $\mathscr{CN} - Tabu$

The objective of repair being to maintain as much as possible the assignments previously performed, it seems intuitively justified to use local search. The current solution s_{cur} is a partial configuration; $\mathscr{CN} - Tabu$ thus appears as a very appropriate approach.

At each iteration, the new configuration is chosen as follows: allocate an un-assigned path by minimizing the number of paths which need to be re-assigned. With this strategy, we expect $\mathscr{CN} - Tabu$ to limit the number of moves and, thus, quickly find a solution close to the previous one. Note however that the objective of maintaining the configuration as close as possible to the previous solution is not directly integrated in the method.

```
Algorithm 4: Rep
Data: $s_{cur}$
Result: $s_{sol}$, nb-rep-min
begin
    nb-rep-min $\leftarrow +\infty$;
    for (seed = 0; seed < nb-runs; seed + +) do
        $s \leftarrow \mathcal{CN}$-Tabu(seed) ;
        if ($|s|$ = nb-paths) then
            nb-rep $\leftarrow$ count-rep($s$);
            if (nb-rep < nb-rep-min) then
                nb-rep-min $\leftarrow$ nb-rep;
                $s^* \leftarrow s$;
    $s_{sol} \leftarrow$ save-sol($s^*$);
    return nb-rep-min;
end
```

Algorithm 4 describes the repair procedure. The main loop calls the $\mathscr{CN} - Tabu$ metaheuristic *nb-runs* times, with different initial random seeds. Given that $\mathscr{CN} - Tabu$ randomly selects one configuration among the best ones identified in the neighborhood, the runs should mostly return different solutions. In our experiments, *nb-runs* is fixed to 20. Notation *nb-paths* indicates the number of paths in the current instance ($2i + 1$ here). Condition $|s| =$ nb-paths is thus used for checking whether s is a complete solution or not; in the first case, it is compared with the best solution found so far. This comparison resorts to procedure *count-rep*, that counts the number of paths re-assigned in s.

4.4.2. Repair by Branch&Bound

In view of the characteristics of the problem, one can expect that blockings are mainly caused by inadequate assignments in the neighborhood of the new link. A possible approach for repairing a solution would then be to concentrate on this neighborhood and to consider every possible assignment. We propose to enumerate this partial set of solutions using the *Branch&Bound* principle. The paths considered are those connected to the two incident sites of l_i, s_1 and s_2; we call $\mathscr{L}_{\text{path}}$ the set of these paths. Also, a time limit is imposed to avoid undue computing times.

During the search, nodes are evaluated by counting the number of paths which are not reassigned to their initial value. Nodes are pruned when this lower bound exceeds the number of repairs in the incumbent solution. When two solutions with a same number of repairs are found, a second criterion based on site availability is implemented. Measure $avail(s)$ is computed for all sites s connected to s_1 or s_2 with a link. The solution with the maximal global availability is retained.

It should be noted that, though applying the *Branch&Bound* with no restriction would guarantee to find a solution, the *Limited Branch&Bound* may fail. In this case, the $\mathscr{CN} - Tabu$ metaheuristic is called. The complete algorithm is detailed in Algorithm 5.

```
Algorithm 5: Rep-LBB
Data: $s_{cur}$
Result: $s_{sol}$, nb-rep-min
begin
    nb-rep-min $\leftarrow$enum($s_{cur}$);
    if (nb-rep-min = $+\infty$) then
        nb-rep-min $\leftarrow$ Rep($s_{cur}$);
    $s_{sol} \leftarrow$ save-sol($s^*$);
    return nb-rep-min;
end
```

Procedure *enum* corresponds to the exact search method and is described in Algorithm 6. The assignment of every consistent pair of frequencies (f_1, f_2) to l_i is sequentially considered. For each pair, *enum* begins with a propagation step. The *Branch&Bound* is then launched. The iteration ends with the deassignment of l_i and the corresponding propagation. In this algorithm, not allocated paths are symbolized with a frequency value -1.

The propagation steps first perform forward checking mechanisms, through function *propagate-assign (p, old-freq, new-freq)*. When a frequency *new-freq* is assigned to a previously non-assigned path p (*old-freq* $= -1$), consistency is restored by disabling conflicting frequency values for constraint linked paths (i.e., a value -1 is assigned to the frequency of these paths). Conversely, the deassignment of p (*propagate-assign(p, new-freq, −1)*) enables these values again. In a same way, function *propagate-avail*(*site*$_E$(*p*), *site*$_R$(*p*), *old-freq, new-freq*) then updates $avail(s)$ for the two sites s incident to p. This allows to quickly compute the global availability as soon as a new solution is found.

```
Algorithm 6: enum
Data: $s_{cur}$
Result: $s^*$, nb-rep-min
begin
    $p_1 \leftarrow p_{2i}, p_2 \leftarrow p_{2i+1}$;
    nb-rep-min $\leftarrow +\infty$, avail-max $\leftarrow 0$;
    for each couple $(f_1, f_2) \in D \times D$ do
        if $(f_1, f_2)$ is consistent then
            % Propagation step
            propagate-assign$(p_1, -1, f_1)$, propagate-assign$(p_2, -1, f_2)$;
            propagate-avail$(site_E(p_1), site_R(p_1), -1, f_1)$;
            propagate-avail$(site_E(p_2), site_R(p_2), -1, f_2)$;
            % Resolution
            $s \leftarrow$ Branch&Bound$(\mathcal{L}_{path})$;
            if $(|s| =$ nb-paths$)$ then
                nb-rep $\leftarrow$ count-rep$(s)$;
                if (nb-rep < nb-rep-min) then
                    nb-rep-min $\leftarrow$ nb-rep;
                    $s^* \leftarrow s$;
                    $f_1{}^* \leftarrow f_1, f_2{}^* \leftarrow f_2$;
            % Propagation step
            propagate-avail$(site_E(p_2), site_R(p_2), f_2, -1)$;
            propagate-avail$(site_E(p_1), site_R(p_1), f_1, -1)$;
            propagate-assign$(p_2, f_2, -1)$, propagate-assign$(p_1, f_1, -1)$;
    return nb-rep-min;
end
```

Finally, the *Branch&Bound* follows a depth-first search strategy. At each node of the search tree, a path is selected according to its order of arrival in the network. Then, the smallest frequency among the ones consistent with the current configuration is selected. To this end, *Branch&Bound* maintains the consistency using the *propagate* functions, as described above. Before every path assignment, the time consumed is compared with the time limit. If this limit is reached, the algorithm stops. In our experiments, this parameter is set to 300 seconds.

5. Results

Our different approaches are evaluated on a set of 36 scenarios supplied by the CELAR. For every instance, the initial kernel contains 17 links. At the end of the deployment, the Hertzian networks contains between 100 and 600 paths (i.e., between 50 and 300 links), between 28 and 168 sites and up to 14755 constraints. The experiments are carried out on an PC station with a Intel Core Duo 1.83 GHz processor.

5.1. General evaluation of the method

We first concentrate on our general method, where the strategies evaluated are the *Avail* greedy heuristic and the *Rep–LBB* repair scheme.

The results obtained are given in Table 1. Column 1 provides the instance number, the 3 next columns, respectively give the number of paths, constraints and sites. The 6 last columns show the results obtained with domains D_1 and D_2 respectively, in number of blockings (# *Bloc*), number of repairs (# *Rep*) and CPU time consumed to carry out the whole deployment (*Time*).

The first remark that can be drawn is that several instances can be entirely solved without any blocking (6 on D_1 and 4 on D_2). The number of paths is 100 in most cases. Also, the computing time is almost null for these instances. Conversely,

Table 1
Instance characteristics and resolution by domain

Instances	Characteristics			D_1			D_2		
	# Paths	# Cstr	# Sites	# Bloc	# Rep	Time	# Bloc	# Rep	Time
01	300	2497	83	3	6	203	6	16	541
02	300	2493	83	3	11	302	6	13	281
03	300	3096	90	2	5	2	1	1	0
04	300	2633	78	3	6	84	8	11	205
05	300	2447	87	0	0	0	2	2	2
06	300	2717	86	5	8	37	7	7	39
10	100	2001	28	2	2	9	4	4	1
11	100	1793	32	0	0	0	0	0	0
12	100	1758	33	1	2	0	0	0	0
13	100	1775	33	1	1	0	0	0	0
14	100	1839	33	0	0	0	1	4	0
15	100	2103	32	4	15	31	2	4	15
16	100	1868	34	0	0	0	1	1	0
17	100	1866	34	0	0	0	0	0	0
18	100	1796	33	0	0	0	1	1	1
19	100	1899	34	1	2	0	1	1	0
20	200	3885	61	3	10	85	3	14	305
21	200	4003	63	5	17	42	10	19	28
22	200	4017	61	5	29	391	6	18	306
23	200	3880	61	3	53	29	4	24	17
24	200	3887	59	5	55	79	4	6	24
25	200	4021	58	3	103	335	10	25	338
26	200	3990	62	3	23	25	4	7	19
27	200	3894	61	4	34	40	6	19	126
28	200	3790	65	1	4	26	4	16	399
29	200	4169	59	10	51	152	6	16	279
30	600	12,662	158	19	31	22	15	25	48
31	600	12,361	160	13	21	325	13	17	189
32	600	12,779	155	×	×	×	19	50	58
33	600	11,552	168	14	288	102	11	23	50
34	600	11,796	168	12	416	1022	19	37	1103
35	600	12,318	164	14	106	78	23	70	154
36	600	13,585	163	×	×	×	30	76	51
37	600	14,480	152	30	192	186	39	119	519
38	600	13,853	160	22	255	3050	29	113	143
39	600	14,755	160	×	×	×	39	158	184

we did not find any feasible solution for three instances (32, 36 and 39) on domain D_1. These instances are marked with a '×' in the corresponding rows.

Our strategies permit to solve most instances with a rather limited number of blockings. The worst results are obtained on instances 37 and 39 with domain D_2, where 13% of the links trigger a repair (39 out of 300 for each instance). It seems obvious that the number of repairs to be carried out increases with the number of blockings encountered. However, the average number of repairs per blocking varies according to the instance category and to the domain. Table 2 synthesizes the average number of repairs per blocking considering the instance category (rows) and the two domains (columns).

Concerning domain D_2, the average number of repairs is similar for all instance categories. Nevertheless, one notes a small increase with the instance size. However, for domain D_1 the results are very different. The two first categories are almost similar. But for the two last ones, one can observe an important change from an average of about 2 repairs per blocking to an average of 9 or 10 on the largest scenarios. The latter instances seem to be really hard to repair efficiently

Table 2
Average number of repairs per blockings

Instances	D_1	D_2
01–06	2.25	1.67
10–19	2.44	1.5
20–29	9.02	2.88
30–39	10.56	2.90

on this domain. Note that the average number of constraints per site on these instances is around 60 for the third category and around 80 for the last one.

Looking further at Table 1, one can notice that the quality of the results is strongly linked to the number of sites and the number of constraints contained in each scenario. Scenarios 22 and 23 provide a very good example since they have the same number of sites. Scenario 22 contains more constraints than scenario 23 and not surprisingly, more blockings are met. Another example concerns instances 28 and 29. Scenario 29 contains more constraints and less sites than scenario 28, and the resolution method encounters 10 times more blockings on the domain D_1 and 1.5 times more on the domain D_2. A last example concerns instances 37 and 38. Scenario 37 contains more constraints and less sites than scenario 38, and the resolution method encounters approximatively 10 additional blockings on both domains. Thus, when constraints are distributed on a smaller number of sites, the deployment is more likely to meet situations of blocking.

Now we will discuss the time spent for solving the different scenarios. This time is very small, below 600 seconds, for all of the instances except instances 34 and 38. It appears that instance 34 is a very difficult instance at least for our method. Nevertheless, the computational requirements for solving these large instances remain reasonable for a practical use of the method.

The analysis of CPU time reveals that some blockings are more difficult to repair than others. For the same number of encountered blockings, the resolution times are very different. For example, we compare the instances 20 and 23 on D_2. The first one meets 3 blockings and is solved in 305 seconds while the second one meets 4 blockings and is solved in only 17 seconds. Scenarios 33 and 34 present the same characteristic. Instance 34 meets one less blocking than instance 33, with a resolution time 10 times larger on domain D_1 and 20 times larger on D_2.

5.2. Comparison of on-line heuristics

In this section, we propose to compare *Avail* and *Min-Freq* (see Section 4.3). Experiments are still made using *Rep–LBB* as the repair algorithm. Table 3 presents the average results for each instance category. The first column gives the instance category. *Avail* and *Min-Freq* are then compared in terms of number of blockings, number of repairs and CPU time in seconds.

It clearly appears that the *Avail* method improves the results. The number of blockings and repairs has decreased for each category, except the 20–29 category on domain D_2.

Table 3
Avail vs. *Min-Freq*

	Instances	*Avail*			*Min-Freq*		
		# Bloc	# Rep	Time	# Bloc	# Rep	Time
D_1	01–06	2.67	6	104.67	3.83	17.83	49.17
	10–19	0.9	2.2	4	0.8	2.7	3.8
	20–29	4.2	37.9	120.4	4.7	54.7	82.9
	30–39	17.71	187	683.57	22.43	224.71	392.86
D_2	01–06	5	8.33	178	8.83	18	134.67
	10–19	1	1.5	1.7	1.7	3.4	8.3
	20–29	5.7	16.4	184.1	5.2	14.4	191.4
	30–39	23.7	68.8	249.9	27.1	76.1	349.6

Table 4
Kernel assignment impact

	Instances	Cnt kernels			Sol kernels		
		# Bloc	# Rep	Time	# Bloc	# Rep	Time
D_1	01–06	2.67	6	104.67	3.67	8.5	122.17
	10–19	0.9	2.2	4	0.8	1.4	5.9
	20–29	4.2	37.9	120.4	3.6	26.5	176.8
	30–39	17.71	187	683.57	17	200.43	935.29
D_2	01–06	5	8.33	178	5.5	6.5	99.5
	10–19	1	1.5	1.7	0.8	3.1	13.6
	20–29	5.7	16.4	184.1	4.9	9.5	82.5
	30–39	23.7	68.8	249.9	21	54.9	236.1

The computing times are of the same magnitude order for both methods. Indeed, most of the computing time is spent when repairing the solutions. One should then expect the most efficient heuristic (*Avail*) to be faster. However, as noted above, the time needed for repairing is very variable.

5.3. Comparison of kernel assignments

As announced in Section 4.2, we propose to evaluate the impact of kernel assignments by comparing our *Cnt* kernels with feasible kernel assignments obtained from the solution of the complete problem (*Sol* kernels). Table 4 synthesizes the results obtained in average for every instance category and for every domain. The first 2 columns indicate the domain and instance category, the next 3 columns give the results obtained on the *Cnt* kernels and the 3 last columns the ones obtained starting from *Sol* kernels. Computing times do not include the computation of the kernels.

Surprisingly, *Sol* kernels do not systematically give the best results. Although slightly better, the results obtained are rather similar. This tends to show that the initial kernel assignment does not really influence the dynamic deployment.

5.4. Comparison of repair methods

Our objective is to determine the improvements obtained by combining the *Limited Branch&Bound* with $\mathcal{CN}-Tabu$ in the repair process. Hence, we compare *Rep–LBB* and the simple *Rep* method, using the *Avail* heuristic for on-line assignment. Table 5 presents the results obtained using these two repair methods. After giving the domain and the instance categories in the 2 first columns, this table presents the average number of blockings and repairs as well as the solution time for *Rep–LBB* in the 3 next columns and for *Rep* in the 3 last columns.

Although the number of encountered blockings are comparable, we note that the exact method brings a significant reduction of the number of repairs for all instances at the expense of a strong increase of the computing time. But, as we already said, these times remain reasonable for a dynamic deployment of an Hertzian network.

However, a particular result draws our attention to computing times spent on instance category 30–39 using domain D_1. They are almost identical for the two repair methods. On these hard instances, the local search encounters the same difficulties to solve some blockings. The detailed results reveal that instance 38 dramatically increases the average computing time, requiring 4432 seconds for being solved.

Table 5
Exact vs. local search

	Instances	*Rep–LBB*			Rep		
		# Bloc	# Rep	Time	# Bloc	# Rep	Time
D_1	01–06	2.67	6	104.67	3.5	23.83	2.67
	10–19	0.9	2.2	4	0.8	7	1.6
	20–29	4.2	37.9	120.4	4.1	81.9	23.8
	30–39	17.71	187	683.57	19	296.14	676.43
D_2	01–06	5	8.33	178	5.67	20.5	0.33
	10–19	1	1.5	1.7	1.2	4.9	0
	20–29	5.7	16.4	184.1	5.1	21.8	0.4
	30–39	23.7	68.8	249.9	23.8	104.4	12.8

Table 6
Limited branch&Bound vs *Branch&Bound*

	Instances	*Rep–LBB*					*Rep-BB*				
		# Bloc	# Rep	Time	# Bk	# Cnt	# Bloc	# Rep	Time	# Bk	# Cnt
D_1	01–06	2.67	6	104.67	7,823,143.67	0.33	2.67	6	158.5	11,840,417	0.33
	10–19	0.9	2.2	4	262,173	0	0.9	2.2	4	262,173	0
	20–29	4.2	37.9	120.4	7,616,682.4	1.2	4.2	37.9	149.8	9,688,705.4	1.2
	30–39	17.71	187	683.57	15,409,000.43	4.43	17.57	132	2,254.71	133,929,891.4	4
D_2	01–06	5	8.33	178	10,588,449.17	0.17	5.17	7.17	206.67	12,206,253.67	0.17
	10–19	1	1.5	1.7	95,232.9	0	1	1.5	1.7	95,232.9	0
	20–29	5.7	16.4	184.1	10,325,403	0.4	5.7	15.2	1,119.4	62,853,086.9	0.3
	30–39	23.7	68.8	249.9	12,513,957.4	4.1	24.2	68.1	2,592.5	151,603,276.8	4

We now evaluate the loss of quality when limiting to 300 seconds the resolution time of each repair by the *Limited Branch&Bound*. With this aim, we run the *Branch&Bound* restricted to the neighbors of the new link, without time limitation. The obtained results are synthesized in Table 6. To the usual information, we add for each of the two methods, the number of backtracks (#Bk) and the number of calls to $\mathcal{CN}-Tabu$ (#Cnt).

For the 3 first instance categories on D_1 and the first 2 ones on D_2 the number of blockings and repairs are almost the same. In fact, the number of backtracks is increased whereas the number of calls to $\mathcal{CN}-Tabu$ is the same. For the blockings two cases are possible: either *Branch&Bound* finds quickly a solution on the restricted path list, or the bad assignments causing the blocking are further in the network and $\mathcal{CN}-Tabu$ is needed on the global path list. For this first instance set, the small instance size allows a realistic running of *Branch&Bound* (fewer than 300 seconds).

Concerning the last instance categories, with more time, results are a bit improved. Only the largest instances on D_1 benefit from an actual improvement. However, the average execution time grows considerably and it spans between 1000 and 2600 seconds. There are much more backtracks (multiplied by 6–12) but the number of calls to $\mathcal{CN}-Tabu$ is almost the same. Finally, limiting the execution time of the exact method does not damage the quality of the obtained results.

6. Conclusion

The *DFAP* is a complex real-life problem composed by three underlying subproblems. This article analyzes the *DFAP* and presents some methodologies able to tackle each subproblem and quick enough to be used in the operational context. The efficiency of the algorithms developed is evaluated on a set of 36 realistic data instances provided by the CELAR.

The first underlying problem is a classical Constraint Satisfaction Problem that consists in assigning the links of the initial kernel. We solve it with the $\mathcal{CN}-Tabu$ metaheuristic. A comparison of the kernels obtained with consistent kernels extracted from a complete solution, tends to show a short impact of the kernel assignment on the realization of the future deployment. However, one should investigate introducing some evaluation criterion (e.g., the maximization of site availability) when solving this first subproblem. Such policy might provide some help for the future assignments.

The second subproblem concerns the assignment of frequencies when new links arrive. A classical greedy heuristic *Min-Freq* was first implemented. A new greedy algorithm based on site availability is then introduced. This strategy selects the frequencies maintaining the maximal availability on the two incident sites of the new link. Due to the good results obtained with this strategy, more evolved site availability measures are developed in Linhares (2007).

The last problem occurs when a new link cannot be assigned and that repair is needed. Two approaches are proposed: a local search method, namely $\mathcal{CN}-Tabu$, and a *Limited Branch&Bound* algorithm followed by $\mathcal{CN}-Tabu$ in case of fail. Both methods first intensify the search in the area of the blocking link, which helps repairing the solution quickly seeing the local scope of the constraints. Compared to $\mathcal{CN}-Tabu$, the *Branch&Bound* method brings a significant reduction of the number of repairs, with acceptable computing times.

A last comment will underline the behavior of the $\mathcal{CN}-Tabu$ metaheuristic. Extensively used within the process of complete resolution, either to solve a subproblem or to assist an exact method, $\mathcal{CN}-Tabu$ proves to be very robust and helpful for finding good solutions quickly. A possible development of the method in this context, might be to introduce the site availability criterion during the neighborhood evaluation phase. One can expect that $\mathcal{CN}-Tabu$ would then better prepare for the arrival of new links.

Given the promising results obtained during this study, we hope these methods to be integrated in the operational systems of the CELAR soon.

Acknowledgements

This work was supported by the CELAR, with particular thanks to T. Defaix. Thanks also to the Coordenacao de Aperfeicoamento de Pessoal de Nivel Superior (CAPES) and to Thales-Communications. The authors also thank the two anonymous referees for their numerous and very helpful comments and suggestions.

References

Aardal, K.I., Hurkens, C.A.J., Lenstra, J.K., Tiourine, S.R., 2002. Algorithms for radio link frequency assignment: The CALMA project. Operations Research 50 (6), 968–980.

Aardal, K.I., van Hoesel, C.P.M., Koster, A.M.C.A., Mannino, C., Sassano, A., 2003. Models and solution techniques for the frequency assignment problem. 4OR 1 (4), 261–317.

Aardal, K.I., van Hoesel, C.P.M., Koster, A.M.C.A., Mannino, C., Sassano, A., 2007. Models and solution techniques for frequency assignment problems. Annals of Operations Research 153, 79–129.

Coffman, E.G., Garey, M.R., Johnson, D.S., 1983. Dynamic bin packing. SIAM Journal of Computing 12, 227–258.

Crescenzi, P., Gambosi, G., Penna, P. 2000. On-line algorithms for the channel assignment problem in cellular networks. In: Proceedings of ACM DIALM'00, International Workshop on Discrete Algorithms and Methods for Mobile Computing, pp. 1–7.

Daniels, K., Chandra, K., Liu, S., Widhani, S., 2004. Dynamic channel assignment with cumulative co-channel interference. ACM SIGMOBILE Mobile Computing and Communications Review 8 (4), 4–18.

Dupont, A., Alvernhe, E., Vasquez, M., 2004. Efficient filtering and tabu search on a consistent neighbourhood for the frequency assignment problem with polarisation. Annals of Operations Research 130, 179–198.

Elkhyari, A., Guéret, C., Jussien, N. 2004. Constraint programming for dynamic scheduling problems. In: Proceedings of ISS'04 International Scheduling Symposium, pp. 84–89.

Fitzpatrick, S., Janssen, J., Nowakowski, R., 2004. Distributive online channel assignment for hexagonal cellular networks with constraints. Discrete Applied Mathematics 143 (1–3), 84–91.

Fleurent, C., Ferland, J.A., 1996. Genetic and hybrid algorithms for graph coloring. Annals of Operations Research 63, 437–461.

Fotakis, D., Pantziou, G., Pentaris, G., Spirakis, P., 1999. Frequency assignment in mobile and radio networks. DIMACS Series on Discrete Mathematic and Theoritical Computer SCIENCE 45. In: Networks in Distributed Computing. AMS, pp. 73–90.

Gendreau, M., Guertin, F., Potvin, J.-Y., Taillard, E., 1999. Parallel tabu search for real-time vehicle routing and dispatching. Transportation Science 33 (4), 381–390.

Grove, E.F. 1995. Online bin packing with lookahead. In: Proceedings of the Sixth Annual ACMSIAM Symposium on Discrete Algorithms, pp. 430–436.

Halldorsson, M.M., Szegedy, M. 1992. Lower Bounds for On-line Graph Coloring. In: SODA'92: Proceedings of the Third Annual ACM-SIAM Symposium on Discrete algorithms, pp. 211–216.

Hertz, A., Schindl, D., Zufferey, N., 2005. Lower bound and tabu search procedures for the frequency assignment problem with polarization constraints. 4OR 3 (2), 139–161.

Kouvelis, P., Yu, G., 1997. Robust Discrete Optimization and its Applications. Kluwer Academic Publishers, Boston.

Linhares, A.C. 2007. ProblFmes d'Affectation de FrTquences Dynamique. Ph.D. thesis, UniversitT d'Avignon et des Pays du Vaucluse. Supported by Coordenatpo de Aperfeitoamento de Pessoal de Nfvel Superior (CAPES) and Faculdades Lourento Filho (FLF).

Palpant, M., Artigues, C., Michelon, P. 2002. A heuristic for solving the frequency assignment problem. In: XI Latin-Iberian American Congress of Operations Research (CLAIO), pp. 27–31.

Palpant, M., Oliva, C., Artigues, C., Michelon, P., Didi Biha, M., 2008. Models and methods for frequency allocation with cumulative interference constraints. International Transactions in Operational Research, 45–58.

Psaraftis, H.N., 1988. Vehicle Routing: Methods and Studies. Elsevier Science Publishers BV, North-Holland, pp. 223–248 (Chapter Dynamic Vehicle Routing Problems).

Vasquez, M., Dupont, A., Habet, D., 2005. Consistent neighbourhood in a tabu search. Metaheuristics: Progress as Real Problem Solvers. MIC-Kluwer, pp. 367–386, Chapter 17.

Vishwanathan, S. 1990. Randomized on-line graph coloring. In: Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science. pp. 121–130.