The Dynamic Frequency Assignment Problem

Audrey Dupont, Andréa Carneiro Linhares, Christian Artigues, Dominique Feillet, Philippe Michelon, Michel Vasquez

The Dynamic Frequency Assignment Problem

Audrey Dupont[1*] Andréa Carneiro Linhares[1] Christian Artigues[2]
Dominique Feillet[1] Philippe Michelon[1] Michel Vasquez[3]

[1]Laboratoire d'Informatique d'Avignon, Université d'Avignon,
Agroparc BP 1228, 84911 Avignon Cedex 9, France
{Audrey.Dupont,Andrea.Linhares,Dominique.Feillet,
Philippe.Michelon}@univ-avignon.fr

[2]LAAS CNRS
7 avenue du Colonel Roche, 31077 Toulouse, Cedex 4, France
artigues@laas.fr

[3]Centre de recherche LGI2P, Ecole des Mines d'Alès
Site EERIE, Parc Scientifique Georges Besse, 30035 Nîmes, Cedex 01, France
Michel.Vasquez@ema.fr

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 deployment. The problem decomposes in three phases: assignment of an initial kernel of communications, dynamic assignment of new communication links and reparation when no assignment is possible. Different solution methods are proposed and many computational tests are carried out on realistic instances.

Keywords: Frequency Assignment, Dynamic Problem, Heuristics, Tabu Search and Consistent Neighborhood, Branch and 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

*Corresponding author. Tel: +33 490 843 518; Fax: +33 490 843 501

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 problem. 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 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 until no frequency can be assigned to a link and avoid interferences. We call this last situation a deadlock. In case of deadlock, a third problem occurs. One has to reassign frequencies so that a consistent assignment is recovered. Several objectives can be pursued. The retained one consists in minimizing the number of reallocated paths.

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

After setting more precisely the problematic in Section 2, we give a detailed description of the problem in Section 3. The different methodologies used to solve the three problem components are then described in Section 4. Finally, Section 5 evaluates these methodologies on 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 testbed for the development and comparison of various optimization strategies and methods. Results were compiled in several survey papers (Aardal *et al.*, 2002; Aardal *et al.*, 2003).

The CELAR then proposed to study more realistic modelings. A first one includes antenna orientations and the possibility to progressively relax interference constraints. This latter aimed at accepting assignments with a slight level of interference, when necessary. 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).

A second model proposes 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).

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

- the data are entirely accessible but with an imprecision;
- the data are 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:

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

In these two cases of uncertainty, a situation where the obtained solutions became unacceptable can arise, which results in failure. It is then necessary to consider the reappraisal of the previous decisions (for example, those taken by the online 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 were widely studied within the framework of various optimization problems, such as Bin Packing (Coffman *et al.*, 1983; Grove, 1995), Scheduling (Elkhyari

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

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 online 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 that the one considered in our study.

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 to assign a frequency to each of its paths.

Formal description needs 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);
- $L = \{l_i\}, \forall i \in \{0, \ldots, n-1\}$, the set of links;
- $P = \{p_i\}, \forall i \in \{0, \ldots, 2n-1\}$, the set of paths.

All the paths of P have the same frequency domain D and two different domains are investigated. The first one D_1 is made up of six intervals: $D_1 = [40000, 40140] \cup [41000, 41140] \cup [42000, 42140] \cup [43000, 43140] \cup [44000, 44210] \cup [45000, 45210]$; 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 by an unique interval $D_2 = [40000, 41890]$, also with a frequency gap of 70. D_2 contains 28 frequency values (see Figure 1). In the following, we note D the domain, which can be either D_1 or D_2.

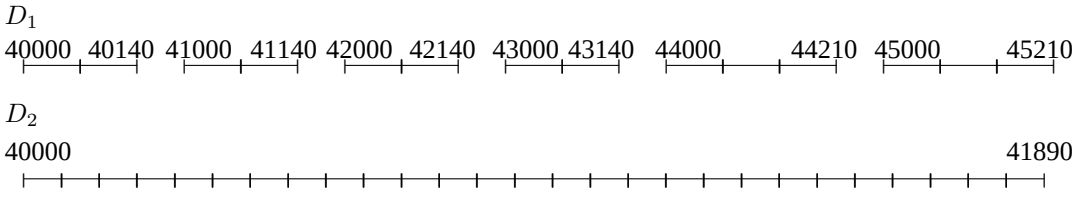

Figure 1: Domain Characteristics

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 closed enough for causing interferences, a binary constraint C_{ij} enforcing some gap between p_i and p_j is defined: $|f_i - f_j| \geq g_{ij}$, where f_i is the frequency of path p_i, f_j the frequency of p_j and g_{ij} the gap required.

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 g_{ij} = 220;
  - the *transmitter-transmitter* constraints, where the gap g_{ij} = 100;
  - the *receiver-receiver* constraints, where the gap g_{ij} is generally closed to 70;
- the far-field constraints, where the gap g_{ij} practically never exceeds 70.

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

Figure 2 presents a small example illustrating the notion of sites, antennas, communication links and the different constraint types.

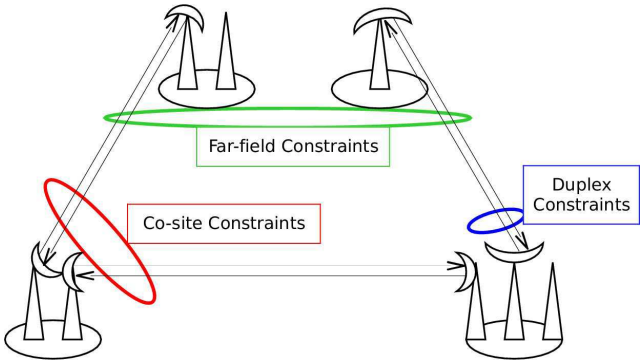

Figure 2: Example of a deployment

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 online 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 online fashion. This defines the second subproblem. In case of deadlock, 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 paths repaired 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.

3.2 Modelling

The *DFAP* can be decomposed into three underlying subproblems, described below.

Static Model

This model corresponds to the frequency assignment for the initial kernel:

$$\text{find } f_0, \ldots, f_{2k-1} \text{ such that:}$$

$$\begin{aligned} |f_i - f_j| &\geq g_{i,j} \qquad (p_i, p_j \in \{p_0, \ldots, p_{2k-1}\}) \\ f_i &\in D \qquad (p_i \in \{p_0, \ldots, p_{2k-1}\}) \end{aligned}$$

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

Dynamic Model

This second model corresponds to the online situation. For each arrival of a new link $l_i = (p_{2i}, p_{2i+1}) \in L$, the problem to solve is:

$$\text{find } f_{2i}, f_{2i+1} \text{ such that:}$$

$$\begin{aligned} |f_{2i} - f_j| &\geq g_{2i,j} \qquad (p_j \in \{p_0, \ldots, p_{2i-1}\}) \\ |f_{2i+1} - f_j| &\geq g_{2i+1,j} \qquad (p_j \in \{p_0, \ldots, p_{2i-1}\}) \\ |f_{2i+1} - f_{2i}| &\geq g_{2i+1,2i} \\ f_{2i}, f_{2i+1} &\in D \end{aligned}$$

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

Repair Model

This last model corresponds to the repair process. For each blocking situation on a link $l_i = (p_{2i}, p_{2i+1}) \in L$, the problem to optimize is:

$$\text{Min}(|\{h \in \{0, \ldots, 2i-1\} / f'_h \neq f_h\}|) \text{ such that:}$$

$$\begin{aligned} |f_{2i} - f'_j| &\geq g_{2i,j} \qquad (p_j \in \{p_0, \ldots, p_{2i-1}\}) \\ |f_{2i+1} - f'_j| &\geq g_{2i+1,j} \qquad (p_j \in \{p_0, \ldots, p_{2i-1}\}) \\ |f'_h - f'_j| &\geq g_{hj} \qquad (p_h, p_j \in \{p_0, \ldots, p_{2i-1}\}) \\ f_{2i}, f_{2i+1} &\in D \\ f'_h &\in D \qquad (p_h \in \{p_0, \ldots, p_{2i-1}\}) \end{aligned}$$

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 specificities of this problem lead to consider three different methodologies with intermediate objectives, to solve each underlying problem: kernel assignment, online assignment and repair process. Different methods are proposed for each of them.

4.1 Kernel Assignment

To assign frequencies to the initial links belonging to the kernel, we propose to use the metaheuristic $\mathcal{CN}$-$Tabu$ presented in (Vasquez *et al.*, 2005). It is an original hybrid Tabu Search algorithm, which deals with partial consistent configurations instead of complete and generally inconsistent ones, like the classical local search methods. So, it computes a consistent neighborhood in a search space defined only by the partial configurations. To do this efficiently, local consistency is maintained by a fast constraint propagation made by incremental techniques on specific data structure proposed in (Fleurent and Ferland, 1996). Thus, $\mathcal{CN}$-$Tabu$ runs on the kernel links to give the kernel assignments (call it *Cnt kernel* hereafter).

In order to evaluate the quality of these allocations, we propose to compare them with reference assignments. They are extracted from a solution of the global network obtained by considering statically the whole deployment. These kernel allocations are called *Sol kernel* and used as references because we are sure that they are consistent: they can be extended to a complete solution of the entire problem.

4.2 Online Assignment

For the online assignment phase, the context imposes to use algorithms fixing the decisions to be made in a sequential way, and forbidding any modification of the previous decisions. Such methods are generally called *greedy algorithms*. In an online 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 incrementally every new assignment request (establishment of a new link), as long as 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. A soon as a solution is found, the greedy algorithm goes back to assign the new links dynamically.

4.2.1 Classical Greedy

Many strategies have been implemented 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...).

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 online algorithm necessitates to assign two frequencies. It is implemented as follows: at each iteration, the algorithm selects the consistent pair of frequency with the minimal highest value. In case of equality, the smallest value is also minimized. Algorithm 1 details this implementation.

Algorithm 1: *Min-Freq*(p_{2i}, p_{2i+1})
begin
  $f_1^* \leftarrow +\infty$, $f_2^* \leftarrow +\infty$, found $\leftarrow false$;
  for *each couple* $(f_1, f_2) \in D \times D$ **do**
    if (f_1, f_2) *is valid* **then**
      if $(\max(f_1, f_2) < \max(f_1^*, f_2^*))$ **then**
        $f_1^* \leftarrow f_1$, $f_2^* \leftarrow f_2$, found $\leftarrow true$;
      else
        if $(\max(f_1, f_2) = \max(f_1^*, f_2^*))$ *and* $(\min(f_1, f_2) < \min(f_1^*, f_2^*))$ **then**
          $f_1^* \leftarrow f_1$, $f_2^* \leftarrow f_2$, found $\leftarrow true$;
  if *found* $= true$ **then**
    $f_{2i} \leftarrow f_1^*$, $f_{2i+1} \leftarrow f_2^*$;
end

In Algorithm 1, f_1 and f_2 represent frequency values tested for f_{2i} and f_{2i+1}; the best pair is stored in (f_1^*, f_2^*). The marker *found* indicates whether a valid frequency pair has been found for the new link or not.

4.2.2 Availability Greedy

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

Roughly, site availability is a measure of the potential of 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 in 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 the classical greedy heuristics presented above, the problem is not handled through an aggregated measure of occupation of the domain, but rather considers the domain for each site separately. Presenting the algorithm necessitates 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, respectively, 220 and 100; 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 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):

$$\begin{aligned} avail_E(s) &= \sum_{f \in D} avail_E(s, f) \\ avail_R(s) &= \sum_{f \in D} avail_R(s, f) \end{aligned}$$

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 a tendency of the network capability of receiving new links in the future, but that its value does not indicate the number of new links that could be added.

To illustrate the notion of availability, Figure 3 shows an example of a site s with the domain D_1. Frequencies used by paths in emission (resp. in reception) are mentioned by letters E (resp. R). Frequencies f available in emission (i.e. such that $avail_E(s, f) = 1$) are indicated by a a.

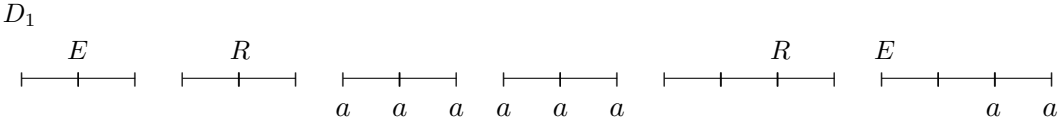

Figure 3: Example of a site s with $avail_E(s)$ = 8

The greedy heuristic is defined as the classical greedy heuristic of Subsection 4.2.1, replacing the minimal frequency criterion with the site availability measure. In case of equivalent availability, the minimal frequency criterion is used. This algorithm is described in Algorithm 2. Note that $avail(f_1, f_2, s)$ 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 also that the test evaluating the validity of (f_1, f_2) is based on

the real values of the constraints (and include all types of constraints) since these values are known for the new link.

Algorithm 2: *Avail*(p_{2i}, p_{2i+1})
begin
  $f_1^* \leftarrow +\infty$, $f_2^* \leftarrow +\infty$, $avail^* \leftarrow -\infty$, found $\leftarrow false$;
  for *each couple* $(f_1, f_2) \in D \times D$ **do**
    if (f_1, f_2) *is valid* **then**
      $avail \leftarrow avail(f_1, f_2, s_1) + avail(f_1, f_2, s_2)$;
      if $avail > avail^*$ **then**
        $f_1^* \leftarrow f_1$, $f_2^* \leftarrow f_2$, found $\leftarrow true$;
      else
        if $(avail = avail^*)$ **then**
          if $(\max(f_1, f_2) < \max(f_1^*, f_2^*))$ **then**
            $f_1^* \leftarrow f_1$, $f_2^* \leftarrow f_2$, found $\leftarrow true$;
          else
            if $(\max(f_1, f_2) = \max(f_1^*, f_2^*))$ *and* $(\min(f_1, f_2) < \min(f_1^*, f_2^*))$ **then**
              $f_1^* \leftarrow f_1$, $f_2^* \leftarrow f_2$, found $\leftarrow true$;
  if *found* $= true$ **then**
    $f_{2i} \leftarrow f_1^*$, $f_{2i+1} \leftarrow f_2^*$;
end

Another promising idea trying to anticipate the future deployment has been tested, with not very convincing results. A set of virtual links was added to the network, around the new link, so as to simulate a possible future development of the network in this area. Characteristics of the set of virtual links were defined according to the characteristics of the existing network (constraint gaps, density...). The criterion chosen to select the frequencies assigned to the new link was then to limit the number of frequencies invalidated in the virtual path domains.

4.3 Repairs

The repair process must re-assign a minimal number of paths in $\{p_0, \ldots, p_{2i-1}\}$, when the new link $l_i = (p_{2i}, p_{2i+1})$ can not be allocated. In the following, s_{cur} is the current solution i.e. a vector of paths and their allocated frequencies ($< (p_0, f_0), ..., (p_{2i-1}, f_{2i-1}) >$) before blocking. The repair process works on configurations which are partial solutions (i.e. not all the variables are assigned).

4.3.1 Repair by $\mathcal{CN}$-$Tabu$

The causes of deadlock are generally bad assignments of the neighboring paths. So, the repair process must be as local as possible, which intuitively justifies the use of local search. We implement the $\mathcal{CN}$-$Tabu$ method which works on partial consistent configurations as stated in Section 4.1. At each iteration, the neighboring configuration is chosen as follow: allocate a non yet assigned variable by minimizing the variable number which need to be re-assigned. This corresponds exactly to the repair process objective. With this space

exploring strategy, we expect that $\mathcal{CN}$-$Tabu$ quickly finds a solution closed to the previous assignments.

Algorithm 3: *Rep*
Data: s_{cur}, l_i
Result: s_{sol}, $nb\text{-}rep$
begin
    nb-rep-min $\leftarrow 2i + 1$;
    for (j = *0; j* $<$ *nb-runs; j* $++$) **do**
        $s \leftarrow \mathcal{CN}$-$Tabu$(*j*) ;
        if ($|s|$ = $2i + 1$) **then**
            nb-rep $\leftarrow$ *eval-rep*(s);
            if (*nb-rep* $<$ *nb-rep-min*) **then**
                nb-rep-min $\leftarrow$ *nb-rep*;
                $s_{sol} \leftarrow$ *save-sol*();
    return *nb-rep-min*;
end

Algorithm 3 describes the repair procedure. The main loop calls *nb-runs* times the $\mathcal{CN}$-$Tabu$ metaheuristic, with different initial random seeds (represented by j), and retains the best found solution. Given that $\mathcal{CN}$-$Tabu$ selects randomly one configuration among the best identified by the neighborhood evaluation, all the runs return different solutions. In our tests, *nb-runs* is fixed to 20. In this algorithm the notation *nb-rep-min* represents the best found *nb-rep*, *s* is the solution returned by $\mathcal{CN}$-$Tabu$ and *s-sol* is the best solution.

However, the experimental results obtained by this approach are not enough convincing. We thus propose another way to repair the blocking situation.

4.3.2 Repair by *Branch&Bound*

An exact method could be used to solve the repair problem. However, exact methods are too much consuming time to be used in an online context. To remain realistic, we propose to limit this exact search following two ways. Firstly a fixed computing time is allowed and secondly the method is run only on a variable subset. Hence, regarding the causes of blocking, we have implemented a *Branch&Bound* algorithm running only on the paths belonging to s_1 and s_2 the two incident sites of l_i.

Algorithm 4: *Rep-LBB*
Data: s_{cur}, l_i
Result: s_{sol}, $nb\text{-}rep$
begin
    nb-rep $\leftarrow$*enum*(s_{cur}, l_i);
    if (*nb-rep* $= 2i + 1$) **then**
        nb-rep $\leftarrow$ *Rep*(s_{cur}, l_i);
    $s_{sol} \leftarrow$ *save-sol*();
    return *nb-rep*;
end

During this search, the solutions found are evaluated by their number of paths which are not re-assigned to there original values. Maintaining this born allows to jump to the next branch as soon as the current configuration has more repair than this upper bound. In case of equality, a second criterion based on the site availability is implemented. In a way similar to *Avail* (see Section 4.2.2), the global availability of all the sites concerned by the repair (i.e. all the neighboring sites of s_1 and s_2) is maximized.

However, it may arise that the bad assignments causing the deadlock are further in the network. In this case, the metaheuristic $\mathcal{CN}$-$Tabu$ assists the exhaustive method *Branch&Bound*. This is detailed in Algorithm 4. Starting from the current configuration s_{cur} and the new link l_i, it calls first the exact search named *enum* and described in Algorithm 5 below. If *enum* does not find a solution, it calls the *Rep* process presented in Algorithm 3.

The method *enum* presented in Algorithm 5 enumerates all the possible assignments for the new link l_i, and for each consistent one launches the *Branch&Bound* on $\mathcal{L}_{path}$ the restricted list of paths which can be repaired.

Algorithm 5: enum
Data: s_{cur}, l_i
Result: s_{sol}, *nb-rep-min*
begin
  $p_1 \leftarrow p_{2i}$, $p_2 \leftarrow p_{2i+1}$;
  nb-rep-min $\leftarrow 2i+1$, *avail-max* $= 0$;
  for *each couple* $(f_1, f_2) \in D \times D$ **do**
    if (f_1, f_2) *is valid* **then**
      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$(site_E(p_2), site_R(p_2), -1, f_2)$;
      nb-rep $\leftarrow 0$;
      Branch&Bound$(\mathcal{L}_{path}, 0)$;
      if (*nb-rep* $<$ *nb-rep-min*) **then**
        nb-rep-min $=$ *nb-rep*;
        ${f_1}^* \leftarrow f_1$, ${f_2}^* \leftarrow f_2$;
      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*$(p_1, f_1, -1)$;
  $s_{sol} \leftarrow$ *save-sol*$()$;
  return *nb-rep-min*;
end

A forward checking mechanism is implemented and performed by the *propagate-assign(path, old-freq, new-freq)* function. To indicate that a path is not yet allocated, it is assigned to -1. Hence, for propagating the instantiation of a path p_i, its *old-freq* is -1 and its *new-freq* is f_i. In opposite the de-assignment of p_i is propagated from f_i to -1. Hence, *propagate-assign* disables or enables the frequency values of the neighboring paths which are in conflict with f_i.In the same way, the *propagate-avail(*$site_E$*(path),* $site_R$*(path), old-freq, new-freq)* function propagates the path assignments and de-assignments on the site availabilities. This

allows to compute quickly the global availability as soon as a new solution is found by the *Branch&Bound*.

Algorithm 6: *Branch&Bound*
Data: $\mathcal{L}_{path}$, nb
Result: s_{sol}
begin
  sol:
  if $(nb = |\mathcal{L}_{path}|)$ **then**
    if (*nb-rep* $<$ *nb-rep-min*) **then**
      $s_{sol} \leftarrow$ *save-sol*();
      nb-rep-min $\leftarrow$ *nb-rep*, *max-avail* $=$ *evaluate-avail*$(\mathcal{L}_{site})$;
    if (*nb-rep* $=$ *nb-rep-min*) **then**
      avail = *evaluate-avail*$(\mathcal{L}_{site})$;
      if (*avail* $>$ *max-avail*) **then**
        $s_{sol} \leftarrow$ *save-sol*(), *max-avail* $\leftarrow avail$;
    $p_i \leftarrow p_{nb-1}$;
    propagate-avail$(site_E(p_i), site_R(p_i), f_i, -1)$;
    propagate-assign$(p_i, f_i, -1)$;
  if (*nb-rep* $>$ *nb-rep-min*) **then**
    $nb--$;
    goto bktk;
  next:
  if (*clock*() $<$ *clock-max*) **then**
    for $(j = p_i.rerun + 1; j < |D|; j++)$ **do**
      if *(*$f_i = D[j]$ *is valid*) **then**
        propagate-assign$(p_i, -1, f_i)$;
        propagate-avail$(site_E(p_i), site_R(p_i), -1, f_i)$;
        $p_i.rerun \leftarrow j$, $nb{+}{+}$;
        if $(nb = |\mathcal{L}_{path}|)$ **then**
          goto sol;
        $p_i \leftarrow p_{i+1}$;
        Branch&Bound$(\mathcal{L}_{path}, nb)$;
        if $(f_i \neq -1)$ **then**
          propagate-avail$(site_E(p_i), site_R(p_i), f_i, -1)$;
          propagate-assign$(p_i, f_i, -1)$;
          $nb--$;
          goto next;
        else
          propagate-avail$(site_E(p_i), site_R(p_i), f_i, -1)$;
          propagate-assign$(p_i, f_i, -1)$;
  bktk:
  $p_i.rerun \leftarrow 0$;
  $p_i \leftarrow p_{i-1}$;
end

Finally, Algorithm 6 presents the recursive version of a *Branch&Bound* algorithm. $\mathcal{L}_{path}$ is the restricted path list and nb is the current number of assigned paths in $\mathcal{L}_{path}$. Hence, a solution is obtained when $nb = |\mathcal{L}_{path}|$ since all the paths are assigned.

Remember that the selection criterion is of two levels. We look at first the number of repairs, since it is the objective of this problem. However, in the case of equality on this criterion, the heuristic aiming at maximizing the availabilities of the concerned sites is added. Thus, $\mathcal{L}_{site}$ is the list of all the incident sites to paths belonging to $\mathcal{L}_{path}$. The solution is chosen if it presents the best repair number, or in case of equality, if the site availability of $\mathcal{L}_{site}$ is maximal.

In this algorithm, p_i is the current branching path of $\mathcal{L}_{path}$. Its *rerun* attribute memorizes the current branching index, in order to restart the search at the next index.

The algorithm has a time limitation. Before each recursive call to *Branch&Bound*, the consumed time is compared to $clock\text{-}max$. If $clock\text{-}max$ is reached, the resolution stops. In our experiments $clock\text{-}max$ is fixed to 300 seconds.

5 Results

Now we will present the results obtained on the set of 36 scenarios supplied by the CELAR. At the end of the deployment, the corresponding Hertzian networks contain between 100 and 600 paths (i.e. between 50 and 300 links), between 28 and 168 sites and up to 14755 constraints. The running tests were carried out on an PC station with a Intel Core Duo 1.83 GHz processor.

The obtained results as well as the main instance characteristics are given in Table 1. Column 1 gives the instance name, the 4 next columns give the path number, the constraint number, the site number and the number of site *Cart 8*, respectively. The 6 last columns present the results obtained with the two domains D_1 and D_2, in terms of number of blockings (*# Bloc* columns), number of repairs (*# Rep* columns) and the CPU time consumed to carry out the whole deployment (*Time* columns), respectively.

The first remark concerns the instances which can be completely solved without deadlock (6 on D_1 and 4 on D_2). They mainly belong to the smallest instances and are instantaneously solved (in 0 seconds).

In opposite, three instances (32, 36 and 39) are not entirely feasible on domain D_1. So, we have not considered their resolution and we set a 'x' in the corresponding rows.

A global tendency shown by Table 1 is that the difficulty of assigning frequencies is relatively proportional to the size of the instances (in terms of blockings, repairs and computing times). Especially, within each category of instances, the number of blocking situations and the number of repairs are rather homogeneous.

	Characteristics			D_1			D_2		
Inst	# 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	12662	158	19	31	22	15	25	48
31	600	12361	160	13	21	325	13	17	189
32	600	12779	155	x	x	x	19	50	58
33	600	11552	168	14	288	102	11	23	50
34	600	11796	168	12	416	1022	19	37	1103
35	600	12318	164	14	106	78	23	70	154
36	600	13585	163	x	x	x	30	76	51
37	600	14480	152	30	192	186	39	119	519
38	600	13853	160	22	255	3050	29	113	143
39	600	14755	160	x	x	x	39	158	184

Table 1: Instance Characteristics and resolution by Domain

Looking further at each instance category, 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 give a very good example. since they have the same number of sites. Scenario 22 contains more constraints than scenario 23 and in a rather natural way, more blocking situations are met. Another example concerns the instances 28 and 29. Scenario 29 contains more constraints and less sites than scenario 28, and the resolution method encounters 10 times more deadlocks 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 deadlocks on the two domains. Thus, the more constraints are distributed on a smaller number of sites, the more the deployment is likely to meet situations of blocking.

It seems obvious that the number of repairs to be carried out increases with the number of encountered blockings. However, the average number of repairs by deadlock varies according to the instance category and to the domain. We propose to look at the synthesized Table 2. It presents the average number of repairs made for each deadlock considering the instance category (rows) and the two domains (columns).

Inst	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

Table 2: Average of number of repairs by blocking situation

Concerning domain D_2, the average number of repairs is similar for all the 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 for an average of about 2 repairs by blocking to an average of 9 or 10 on the greatest scenarios. These last instances seem to be really hard to repair efficiently on the divided domain. Note that the average of constraints by site on these instances is around 60 for the third category and around 80 for the last one.

Now we will discuss on the time spent for solving the different scenarios. In a general way, it is very small, below 600 seconds for each instance except two (around 1000 seconds for instance 34 on the two domains, and up to 3000 seconds for instance 38 on D_1). It appears that instance 34 is a very difficult instance at least for our solving method. In spite of that, the computational requirements remain reasonable for solving these large instances.

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

5.1 Comparison of Online Heuristics

In this section, we propose to compare the *Avail* heuristic with the simpler one *Min-Freq* (see Section 4.2). Table 3 presents the results in average on each instance category depending on the path number. The first column gives the instance category (for example, 01-06 concerns the instances from 01 to 06). *Avail* and *Min-Freq* are then compared in terms of numbers of blocking/repair and CPU time in seconds.

		Avail			*Min-Freq*		
	Inst	# 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 3: *Avail* vs. *Min-Freq*

It appears clearly 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.

On domain D_1, this improvement yields an increase of the computing time, which however remains very acceptable. For example, the mean solution time of the largest instances is less than 700 seconds, which is quite convenient in practice seeing that this corresponds to assigning iteratively 300 links and calling 18 times the repair method.

Concerning domain D_2, the results are different. *Min-Freq* takes more time for solving the three last instance categories whereas *Avail* gives better results for three of the four categories. This can be surprising since propagating the availabilities on top of assignments should take more time. In fact, *Min-Freq* and *Avail* resemble exact search limited to the two news paths. For domain D_2 the *Avail* criterion avoids to redo quasi similar calculations for couple (d_1, d_2) closed on the continuous domain. This is not the case on the disjoint domain D_1.

5.2 Comparison of Kernel Assignments

As announced in Section 4.1, we propose to evaluate the impact of kernel assignments by comparing our *Cnt* kernels with feasible kernel assignments obtained by solving the complete problem (*Sol* kernels). Table 4 synthesizes the results obtained in average on every instance category and on every domain. The 2 first columns indicate the domain and the instance category, the 3 next 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.

		Cnt Kernels			Sol Kernels		
	Inst	# 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

Table 4: Kernel Assignment Impact

Surprisingly, *Sol* kernels do not give systematically 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.

5.3 Comparison of Repair Methods

Now we want to determine the bringing-in of using an exact method to solve the repair problems. Hence, we propose to compare the *Limited Branch&Bound* with the repair method

Rep using 20 runs of $\mathcal{CN}$-$Tabu$. 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 blocking and repair numbers as well as the solving time for *Rep-LBB* in the 3 next columns and for *Rep* in the 3 last columns.

		Rep-LBB			Rep		
	Inst	# 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 5: Exact vs Local Search

Although the number of encountered blockings are comparable, we note that exact method brings a significant reduction of the repair number on all the instances at the expense of a strong increase of the computational requirements. But, as we already said, these times remain reasonable in the 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. Hence, on these hard instances, the local search encounters the same difficulties to solve some blocking situations. The detailed results reveal that instance 38 dramatically increases the average computing time, requiring 4432 seconds for being solved.

We now evaluate the loss of quality while limiting to 300 seconds the resolution time of each repair by the *Limited Branch&Bound*. In 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 encountered backtracks, as well as the number of calls to metaheuristic $\mathcal{CN}$-$Tabu$. Given that *Branch&Bound* is restricted to the only neighbors of the new link, it may arise that it is not enough to find a solution. It comes to the fact that the paths whose assignments are "blocking", are further in the network. In these cases, the call to $\mathcal{CN}$-$Tabu$ is necessary to find a solution.

		Rep-LBB					*Rep-BB*				
	Inst	# 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

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

For the 3 first instance categories on D_1 and the 2 first ones on D_2 the number of blockings and repairs are quite the same. In fact, the numbers of backtracks are increased but there are too many calls to $\mathcal{CN}$-$Tabu$. For the deadlocks two cases are possible: either *Branch&Bound* finds quickly a solution on the restricted path list, or the blocking causes are further in the constraint network and $\mathcal{CN}$-$Tabu$ is needed on the global path list. On the smallest instances the number of variables, constraints and values in the domain remain realistic to run a *Branch&Bound* in 300 seconds.

Concerning the last instance categories, with more time, results are a bit improved. Only the largest instances benefit from an actual improvement on D_1. However, the average of execution times grows considerably and it ranges between 1000 and 2600 seconds. There are much more backtracks (multiplied by 6 to 12) but the numbers of calls to $\mathcal{CN}$-$Tabu$ are 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 complex real-life problem defining three underlying subproblems that needs to be solved in the Operations Research framework. This article presents a study of methodologies able to tackle each sub-problem.

The first underlying problem consisting in assigning the links belonging to the initial kernel is a classical Constraint Satisfaction Problem. We have proposed to use the $\mathcal{CN}$-$Tabu$ metaheuristic to find quickly the kernel assignments. In a second time, we proposed to compare them with consistent kernels extracted from a global solution. However, due to the small size of the kernels, we can envisage many other assignment methods, even exact methods. This implies to find realistic comparison criteria between solutions.

For dynamic assignment, we have first presented *Min-Freq*, a classical heuristic used in frequency assignment framework. Then we described a greedy algorithm based on the site availability for emission and for reception. This strategy selects the frequencies which let the maximal availability on the two incident sites of the new link. Due to the good results obtained by this strategy, new criteria to measure the site availability are defined in (Linhares *et al.*, 2005).

The last problem concerns the repairs in case of deadlock. According to the objective, the repair must be as local as possible. We have implemented a *Branch&Bound* algorithm limited in two ways. On the first hand, it is restricted to work only on the neighboring paths of the new link and on the second hand a time limitation is fixed to 300 seconds. If no solution is found, we resort to $\mathcal{CN}$-$Tabu$ as a local search method.

Our last remark concerns the $\mathcal{CN}$-$Tabu$ metaheuristic. It is extensively used for two problem resolutions, either to solve the problem or to assist an exact method. Each time, this method is robust and very helpful to give good results very quickly.

Finally, many results are presented to evaluate the impact of the initial kernel assignments, the efficiency of the *Avail* greedy method and the contribution of the *Limited Branch&Bound*.

Acknowledgements

This work was supported by the CELAR, with a particular thanks to T. Defaix. Thanks also to the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) and to Thales-Communications.

References

Aardal, K. I., Hurkens, C. A. J., Lenstra, J. K., and 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., and Sassano, A. 2003. Models and Solution Techniques for the Frequency Assignment Problem. *4OR*, **1**(4), 261–317.

Coffman, E. G., Garey, M. R., and Johnson, D. S. 1983. Dynamic Bin Packing. *SIAM Journal of Computing*, **12**, 227–258.

Crescenzi, P., Gambosi, G., and Penna, P. 2000. On-line Algorithms for the Channel Assignment Problem in Cellular Networks. *Pages 1–7 of: Proc. ACM DIALM'00, International Workshop on Discrete Algorithms and Methods for Mobile Computing*.

Daniels, K., Chandra, K., Liu, S., and 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., and 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., and Jussien, N. 2004 (May). Constraint programming for dynamic scheduling problems. *Pages 84–89 of: Proc. of ISS'04 International Scheduling Symposium*.

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

Fleurent, C., and Ferland, J.A. 1996. Genetic and Hybrid Algorithms for Graph Coloring. *Annals of Operations Research*, **63**, 437–461.

Fotakis, D., Pantziou, G., Pentaris, G., and Spirakis, P. 1999. Frequency Assignment in Mobile and Radio Networks. *DIMACS Series on Discrete Mathematic and Theoritical Computer SCIENCE 45, Networks in Distributed Computing, AMS*, 73–90.

Gendreau, M., Guertin, F., Potvin, J.-Y., and 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. *Pages 430–436 of: Proceedings of the Sixth Annual ACMSIAM Symposium on Discrete Algorithms.*

Halldorsson, M. M., and Szegedy, M. 1992. Lower Bounds for On-line Graph Coloring. *Pages 211–216 of: SODA'92: Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*.

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

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

Linhares, A., Artigues, C., and Feillet, D. 2005 (26-28 October). A strategy based on site availability for dynamic frequency assignment problems. *In: The Fifth ALIO/EURO conference on combinatorial optimization, Paris, France*.

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

Psaraftis, H. N. 1988. *Vehicle Routing: Methods and Studies*. Chap. Dynamic Vehicle Routing Problems, pages 223–248.

Vasquez, M., Dupont, A., and Habet, D. 2005. *Consistent Neighbourhood in a Tabu Search*. Metaheuristics: Progress as real Problem Solvers. MIC-Kluwer. Chap. 17, pages 367–386.

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