HAL open science

Efficient Filtering and Tabu Search on a Consistent Neighbourhood for the Frequency Assignment Problem with Polarisation

Audrey Dupont, Eric Alvernhe, Michel Vasquez

To cite this version:

Audrey Dupont, Eric Alvernhe, Michel Vasquez. Efficient Filtering and Tabu Search on a Consistent Neighbourhood for the Frequency Assignment Problem with Polarisation. Annals of Operations Research, 2004, 130, pp.179-198. 10.1023/B:ANOR.0000032575.38969.ab . hal-00353829

HAL Id: hal-00353829

https://hal.science/hal-00353829v1

Submitted on 17 Oct 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Efficient Filtering and Tabu Search on a Consistent Neighbourhood for the Frequency Assignment Problem with Polarisation

AUDREY DUPONT *, ERIC ALVERNHE and MICHEL VASQUEZ
{dupont;alvernhe;vasquez}@site-eerie.ema.fr
Centre de recherche LGI2P, École des Mînes d'Alès, Site EERIE, 30035 Nîmes Cedex 1, France

Abstract. This article presents a specific filtering algorithm for the Frequency Assignment Problem with Polarisation, which combines arc-consistency and path-inverse-consistency adapted to the specificities of the constraints. The effectiveness of this filtering algorithm enabled us to improve the *Tabu Search* on a Consistent Neighbourhood (C$\mathcal{N}$-Tabu) using two different approaches. So, after a short recall of this general methodology and a presentation of its obtained results on the FAPP, we propose a behavioural study of the two approaches by comparing the results.

Keywords: frequency assignment problem, arc-consistency, path-inverse-consistency, Tabu Search, Consistent Neighbourhood

Introduction

The Frequency Assignment Problem with Polarisation (FAPP) in Hertzian telecommunication networks consists in assigning frequency resources so as to minimise electromagnetic interference due to the proximity of antennae and interactions within the same Hertzian liaison between emitter and receptor frequencies.

To solve the FAPP, a *Tabu Search* method working on consistent partial configurations (C$\mathcal{N}$-Tabu) was implemented. The originality of the approach is that it deals with a consistent neighbourhood defined on these particular configurations.

In order to control the global quality of the Hertzian network, in terms of interference, relaxation levels were introduced on some electromagnetic constraints. Taking into account this specificity, we improved our resolution method by an original filtering process, which combined *Arc-Consistency* (AC) on all the constraints and *Path-Inverse-Consistency* (PIC) on the relaxable constraint cliques.

The solution search, strongly linked to the relaxation level allows to encapsulate the C$\mathcal{N}$-Tabu following two general approaches: the first one starts from the highest level where maximal relaxation is authorized. The second one begins from the last consistent level calculated by the filtering algorithms. Analysis of these approaches emphasises the stability and speed of the Tabu Search on a Consistent Neighbourhood.

* Corresponding author.

This article begins with the modelling of the problem in section 1. Section 2 describes the filtering algorithms, and gives the results obtained. The next part presents the FAPP resolution by detailing the tabu search on a consistent neighbourhood for solving a constraint network, then followed in section 4 by both implementations: *Downward* and *Upward*, and a behavioural study by comparison of their results. Finally, section 5 summarises the results obtained by other methods developed to solve the FAPP.

1. Problem presentation

1.1. Physical description

The FAPP concerns a Hertzian telecommunication network made up of transmission equipment (antennae connected to emitters or receptors) located at a set of geographical sites. A Hertzian liaison joins two sites by one or more paths. Hence, a path is a unidirectional radio-electric bond, established between antennae at distinct sites, which has a given frequency and polarisation. A frequency resource for a path is a pair (frequency, polarisation), whose components are respectively associated to the carrying frequency of the transmitted signal and the wave polarisation. In the simplified model, the polarisation is a binary variable (i.e., only vertical or horizontal). For each path x_i, a set of available resources $\{(f_i,p_i),\ f_i \in F_i \text{ and } p_i \in P_i\}$, where F_i is the ordered frequency domain representing the authorised wave band, and P_i is the polarisation information, which may include a required polarisation.

So, the frequency assignment problem consists in finding, for each path, a frequency resource satisfying the radio-electric compatibility constraints. Moreover, given that many emitters and receptors are located on the same site, other coupling binary constraints linking paths located on the same site, or linking two paths belonging to the same Hertzian liaison must be also verified. Thus, the constraint set is composed by:

1. equality or inequality of frequencies across two paths: $f_i = f_j$ or $f_i \neq f_j$;
2. distance between frequencies of two paths: $|f_i - f_j| = \varepsilon_{ij}$ or $|f_i - f_j| \neq \varepsilon_{ij}$;
3. equality or inequality of polarisation across two paths: $p_i = p_j$ or $p_i \neq p_j$ (not all the paths are linked by this constraint type);
4. minimal distance between frequencies of two paths:

$$|f_i - f_j| \geqslant \begin{cases} \gamma_{ij} & \text{if } p_i = p_j, \\ \delta_{ij} & \text{if } p_i \neq p_j. \end{cases}$$

The last constraint type controls the interference phenomenon, which is why the required distance between frequencies depends on their polarisations: it is smaller if the polarisations are different (i.e., $\delta_{ij} \leqslant \gamma_{ij}$).

One feasible solution is thus to assign all the paths that completely satisfy this constraint set. Unfortunately, most problems do not have feasible solutions, because the domains are too restrictive or the requirements too numerous. Consequently, the

operator must look for a "good quality" solution. This means that some deterioration is allowed, by permitting some interference. Thus, the intention is now to minimise this interference. With this aim, two constraint classes are introduced:

- IC: strong or Imperative Constraints (type 1–3 above);
- ECC: Electromagnetic Compatibility Constraints (type 4), where a progressive relaxation is authorised and expressed by relaxation levels: level 0 corresponds to no relaxation, and going from level k to level $k+1$ involves the relaxation of some or all the frequency gaps, the maximum relaxation level being 10:

$$|f_i - f_j| \geqslant \begin{cases} \gamma_{ij}^0 \geqslant \cdots \geqslant \gamma_{ij}^k \geqslant \cdots \geqslant \gamma_{ij}^{10} & \text{if } p_i = p_j, \\ \delta_{ij}^0 \geqslant \cdots \geqslant \delta_{ij}^k \geqslant \cdots \geqslant \delta_{ij}^{10} & \text{if } p_i \neq p_j, \end{cases}$$

  since in the 11th level, $\gamma_{ij}^{11} = \delta_{ij}^{11} = 0$, so there is no ECC.

So, we note ECC_k the set of ECC constraints at level k; this means that each constraint belonging to ECC_k is affected to its γ_{ij}^k and δ_{ij}^k gaps.

Accordingly, a feasible solution at level k is an assignment of all the paths satisfying all the strong constraints IC and all the ECC_k constraints. If such a solution exists, the problem is said to be k-feasible. Consequently, the objective function of the problem becomes, in order of priority:

1. search the lowest relaxation level k for which a k-feasible solution exists;
2. then, minimise $V^{(k-1)}$: the number of constraints of $\mathrm{ECC}_{(k-1)}$ unsatisfied at level $k-1$;
3. finally, minimise $\sum_{0 \leqslant i < k-1} V^{(i)}$: the sum of the constraints of ECC_i unsatisfied at all levels i less than $k-1$.

Considering the first objective of the evaluation function, the optimal solution s^* will be necessarily at the lowest k-feasible level (k^*).

1.2. Modelling as a constraint network

Setting the paths as variables, frequencies and polarisations as their domain values, and IC and ECC as the constraint set, the FAPP can be formalised as a Maximal Constraint Satisfaction Problem (Max-CSP), on the following constraint network:

- let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be the set of the n paths;
- let F_i be the frequency domain of x_i, and P_i the polarisation domain, where $P_i \in \{\{-1\}, \{1\}, \{-1, 1\}\}$, each resource to be affected to each variable x_i is a pair (f_i, p_i) where $f_i \in F_i$ and $p_i \in P_i$;
- let IC be the set of the imperative constraints: $\mathrm{IC} = \{p_i = p_j, p_i \neq p_j, |f_i - f_j| = \varepsilon_{ij}, |f_i - f_j| \neq \varepsilon_{ij}\}$, ε_{ij} can be null; in this way, all the frequency constraints are merged in two constraint types: Equality of the Distance between two Frequencies (EDF) ($f_i = f_j$ and $|f_i - f_j| = \varepsilon_{ij}$) and Difference of the Distance between two Frequencies (DDF) ($f_i \neq f_j$ and $|f_i - f_j| \neq \varepsilon_{ij}$).

- let ECC_k for $0 \leqslant k \leqslant 10$ be the set of the relaxed constraints:

$$|f_i - f_j| \geqslant \frac{|p_i + p_j|}{2}\gamma_{ij}^{(k)} + \frac{|p_i - p_j|}{2}\delta_{ij}^{(k)};$$

  Every problem is 11-feasible.

The strategy adopted for the resolution consists in transforming the Max-CSP (an optimisation problem) into 11 CSP (decision problems) according to the relaxation level on the electromagnetic compatibility constraints: each CSP(k) contains both the IC and the ECC_k constraints.

This enables us to introduce some filtering treatments on each CSP(k), in order to obtain the k consistent level closest to level k^*. The second advantage of this filtering process is to reduce the domain size, by deleting the inconsistent values.

2. Filtering algorithms for the FAPP

2.1. Basic definitions

After recalling some concepts of consistency, we briefly present two filtering algorithms: *Arc-Consistency* (AC) then *Path-Inverse-Consistency* (PIC).

Let $\mathrm{CN} = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ be a binary constraint network, and $\mathcal{C}(ij)$ a constraint in $\mathcal{C}$ linking x_i and x_j. A value $f_i \in D_i$ is *consistent* with $\mathcal{C}(ij)$ if and only if $\exists f_j \in D_j$ such that $(f_i, f_j) \in \mathcal{C}(ij)$, f_j is then called a *support* for (x_i, f_i) on $\mathcal{C}(ij)$. A value f_i is called *viable* if it has at least one support on each $\mathcal{C}(ij)$.

A domain D_i is called *consistent domain*, if all of its values have at least one support on each constraint.

Introduced by Waltz (1975), *Arc-Consistent* filtering (AC) on a constraint network is achieved by removing all non-viable values. Some variant AC algorithms have been elaborated, for example AC3 (Mackworth, 1977) and AC-inference (Bessière, Freuder, and Régin, 1995).

To refine the filtering operation, Freuder proposes in (Freuder and Elfe, 1996) the *k-inverse consistency*, in which each variable value that cannot be extended to a consistent instantiation including $k - 1$ additional variables is removed. We shall use the *Path Inverse Consistency* (PIC), which is the 3-inverse consistency. More formally, a constraint network is PIC-consistent, if and only if $\forall x_i, x_j, x_k \in \mathcal{X}$ such that $x_j \neq x_i \neq x_k \neq x_j, \forall v_i \in D_i$: $\exists v_j \in D_j$ and $v_k \in D_k$ such that $\{(x_i, v_i), (x_j, v_j), (x_k, v_k)\}$ is locally consistent.

For more details concerning filtering algorithms, the interested reader can refer to (Debruyne and Bessière, 1997).

2.2. General methodology

It may seem unworkable to attempt to use a pre-processing filtering on an optimisation problem. However, as stated in section 1.2, the resolution strategy adopted for the FAPP

transforms the initial optimisation problem into 11 decision problems. Hence, each CSP resolution is preceded by a filtering process.

Starting from level 11 down to the consistent level, the filtering algorithm finds, for each domain value, the minimal level at which the value has, at least one support on each constraint. So, it stops when it empties one variable domain, and returns the level above, which is the last consistent level where a solution can possibly be found. This algorithm is directly inspired by the AC3 and PIC algorithms.

The next three sections present this filtering process. To be precise, for each constraint type $\mathcal{C}(ij)$, they detail the computing of the consistent domain of a variable x_i involved in this constraint type, then the propagation mechanisms (i.e., how to maintain the consistency of the domain D_j linked by $\mathcal{C}(ij)$ to D_i where a value was deleted by an other constraint).

2.3. Arc-consistency on binary constraints

Arc-consistency is carried out on each constraint type. So, we begin with the equality and difference constraints of distance between two frequencies, which are easy to process, then continue with the specific processing of the electromagnetic compatibility constraint type.

2.3.1. Distance constraints: equality and difference

The equality of the distance between frequencies. EDF are expressed by $|f_i - f_j| = \varepsilon_{ij}$, where $\varepsilon_{ij} \in \mathbb{N}$ can be null (i.e., $f_i = f_j$). For checking the D_i consistency, it is enough to check for each $f_i \in D_i$ if $f_i - \varepsilon_{ij}$ or $f_i + \varepsilon_{ij}$ are in D_j. For all the equality constraints between x_i and a neighbouring variable x_j with a gap ε_{ij}, the deletion of f_i from D_i is propagated on D_j by deleting $f_i + \varepsilon_{ij}$ (respectively $f_i - \varepsilon_{ij}$) if $f_i + 2 \times \varepsilon_{ij}$ (respectively $f_i - 2 \times \varepsilon_{ij}$) is not in D_i.

Difference of the distance between frequencies. DDF are expressed by $|f_i - f_j| \neq \varepsilon_{ij}$, where $\varepsilon_{ij} \in \mathbb{N}$ can be null (i.e., $f_i \neq f_j$). Checking the consistency on a difference constraint is easier. Indeed, for each $f_i \in D_i$, all the D_j values different from $f_i - \varepsilon_{ij}$ and from $f_i + \varepsilon_{ij}$ are supports for f_i on $\mathcal{C}(ij)$. Hence, if the domains D_j contains more than two values, then it is sure of finding a f_i support. If D_j has only one value f_1, we delete the two values $f_i = f_1 - \varepsilon_{ij}$ and $f_i = f_1 + \varepsilon_{ij}$ if they belong to D_i. If D_j has two values (f_1 and f_2), we propagate on D_i if and only if $\exists f_i \in D_i$ such as $f_i = f_1 + \varepsilon_{ij} = f_2 - \varepsilon_{ij}$. Consistency domain of D_j is compute in the same way. Concerning the propagation on this constraint type $\mathcal{C}(ij)$, considering that a value f_i was deleted from D_i by a $\mathcal{C}(ik)$ constraint, any propagation is needed on D_j if and only if the filtered D_i has more than two values. Otherwise, we use the same consistency check as to compute the consistent domains.

2.3.2. The electromagnetic compatibility constraints (ECC)

The mathematical formula of this constraint type is $|f_i - f_j| \geqslant \varepsilon_{ij}$, where $\varepsilon_{ij} = \gamma_{ij}$ if polarisations p_i and p_j are equal, and δ_{ij} otherwise.

In the following section, a different reasoning method is adopted to compute the domain consistency. Rather than searching for a support f_j on $\mathcal{C}(ij)$ for each domain value f_i, we search the value set supported by D_j. Indeed, instantiating x_i and x_j consists in respecting one of the two inequalities, $f_i < f_j$ or $f_i \geqslant f_j$. In the first inequality, assigning $\max(D_j)$ to x_j leads to the potential consistent domain for x_i, $[-\infty, \max(D_j) - \varepsilon_{ij}]$. In the second one, assigning $\min(D_j)$ leads to the second potential consistent domain, $[\min(D_j) + \varepsilon_{ij}, +\infty]$. Considering the two inequalities, the full potential consistent domain becomes $Cons(i) = [-\infty, \max(D_j) - \varepsilon_{i,j}] \cup [\min(D_j) + \varepsilon_{i,j}, +\infty]$. However, this interval union is not necessarily a subset of D_i, so the consistent domain is effectively $D_i \cap Cons(i)$. Since the domains are ordered, this algorithm has the same complexity as a two set intersection computation, so it is linear. Concerning the propagation mechanism, no updating is needed even if a value is deleted from the domain D_i while $\min(D_i)$ and $\max(D_i)$ have not changed.

2.4. Path inverse consistency on 3-cliques with 3 ECCs

Path inverse consistency is implemented for the electromagnetic compatibility constraint cliques. As an AC refinement, PIC increases the deleted value number, by working simultaneously on three constraints. This idea was inspired by (Hertz, Schindl, and Zufferey, 2001), but is implemented using our method of consistency computing and propagation on the ECCs.

In fact, stating that arc-consistency is 2-inverse-consistency allows the reasoning to be extended to ECC cliques. At this stage, arc-consistency is carried out on three variables (x_1, x_2 and x_3) linked by three ECCs. So, there are 6 possible inequalities ($f_1 < f_2 < f_3$, $f_1 < f_3 < f_2, \ldots$). Considering any one of them, all the constraints lose their absolute term. In order to compute the arc consistent domain of one variable, we determine the potential consistent interval for each inequality. Let l (where $l \in [1, 3]$) be the variable position in the current order, its potential consistent domain is computed as follows:

- each of the $l - 1$ first variables is assigned to the minimal consistent value according to the constraints linking it to the previous variables in the order;
- each of the $3 - l$ last variables is assigned to the maximal consistent value according to the constraints linking it to the next variables in the inverse order;
- now, we define $[a, b]$ as the potential consistent domain for the lth variable on the current order; a (respectively b) is the minimal (respectively maximal) consistent value for all the constraints linking the $(l - 1)$th first (respectively $(3 - l)$th last) variables; a or b could be $-\infty$, or $+\infty$, or not defined, or any other value (not necessarily in the domain of the lth variable domain);
- if $l = 2$, we check the *crossing constraint* (linking the first and the third variables) before computing a and b.

If one of these points is not defined, the potential consistent interval in the considered order is empty. The consistent variable domain is therefore the intersection between

its domain and the union of the six intervals obtained by the different orders. As the domains are sorted, local consistency checking is linear.

2.5. *Path-inverse-consistency on 3-cliques with 2 ECCs and one EDF*

A 3-inverse-consistency on two ECCs and one EDF is implemented if the distance value of the equality constraint ε_{13} is smaller than the two ECCs values sum: $\varepsilon_{13} < \varepsilon_{12} + \varepsilon_{23}$, where $\mathcal{C}(13)$ is the EDF, $\mathcal{C}(12)$ and $\mathcal{C}(23)$ are the ECCs. Under this condition, the orders $f_1 < f_2 < f_3$ and $f_3 < f_2 < f_1$ entail the violation of at least one constraint.

Hence, finding the consistent domain of x_2 consists in looking for all f_2 the pairs (f_1, f_3), where $f_1 \in D_1$ and $f_3 \in D_3$ must be simultaneously higher, or simultaneously lower than f_2. If they exist, four value pairs (f_1, f_3) are well designed to be supports: $(f_1, f_3 = f_1 - \varepsilon_{13})$ and $(f_1, f_3 = f_1 + \varepsilon_{13})$, set in the left of the domains, then $(f_1 = f_3 - \varepsilon_{13}, f_3)$ and $(f_1 = f_3 - \varepsilon_{13}, f_3)$ set in the right. A value f_2 is consistent if it is supported by one of these pairs. So, calculating the x_2 consistent domain is linear.

Computing a x_1 consistent domain consists in checking for each value f_1 whether $f_1 + \varepsilon_{13}$ or $f_1 - \varepsilon_{13}$ belongs to D_3. Therefore, we deduce the f_1 3-inverse-consistency by verifying if one of these pairs can be extended with one of the x_2 consistent values. Finally, calculating the x_3 consistent domain is similar to x_1.

2.6. *Filtering results*

Experiments were carried out on the FAPP benchmark proposed for the ROADEF'01 challenge. This consists of 40 instances, having 200 to 3000 variables, 163 to 15664 imperative constraints and 945 to 30625 electromagnetic compatibility constraints. The cardinality of the whole domain size is 2 million. Tests were performed on a 1.9 GHz Pentium IV processor.

After giving the instance characteristics: instance name which includes the variable number in column 1, the imperative constraint number (column 2), the ECC number (column 3) and the total size of all the domains (column 4), table 1 shows a comparison between a generic AC3 algorithm (column 5) and our specific AC3 (column 6). Generic AC3 checks all the value pairs, while specific AC3 is adapted to constraint characteristics. Because these two algorithms implement the same filtering, this two versions obviously give the same consistent level k_{const} in column 7, and the same percentage of filtered values according to the initial domain size in the *ratio* column. So, we compare their computation time in seconds in columns 5 and 6.

The AC3 time analysis shows that the specific version is much faster than the generic one, up to 100 times faster in some instances, for example, 12-1500, 14-2500 and 39-2750.

The last part of table 1, presents the results obtained by our PIC only on the particular 3-cliques, as mentioned in sections 2.4 and 2.5, which is why we refer to PIC$^-$ in the results table. The k consistent level is given in column 9, then the ratio of filtered values at this level according to the initial domain size is presented in column 10 and finally the

Table 1
Filtering results.

Instance name	Characteristics			AC3				PIC$^-$		
	IC	ECC	Init dom.	gen.	spec.	k_{const}	ratio	k_{const}	ratio	time
01-0200	163	945	26963	3	1	3	52.85	4	51.85	1
02-0250	217	1419	36618	6	1	2	59.70	2	63.16	1
03-0300	277	2049	53536	7	1	7	47.30	7	54.42	1
04-0300	238	1561	61762	16	1	1	64.44	1	66.40	2
05-0350	311	2177	79311	11	1	8	31.70	8	32.31	1
06-0500	425	3053	108024	22	1	5	50.90	5	56.49	2
07-0600	559	4218	109658	11	1	9	36.21	9	38.01	1
08-0700	546	3288	134020	20	1	5	38.86	5	40.08	2
09-0800	625	4175	121824	17	1	3	56.54	3	58.29	2
10-0900	761	5310	197665	35	1	6	38.25	6	44.88	4
11-1000	978	7027	294634	63	1	8	48.16	8	49.31	3
12-1500	1469	11970	436967	206	2	2	62.32	2	65.19	11
13-2000	1686	11983	320494	68	2	3	54.80	3	56.40	4
14-2500	2453	19157	774322	293	3	4	58.61	4	60.75	20
15-3000	2487	15267	515606	89	2	5	40.63	5	41.07	3
16-0260	249	1839	47293	1	1	11	1.42	11	1.42	1
17-0300	247	1809	64034	1	1	4	98.57	4	98.57	1
18-0350	962	1425	73016	1	1	8	98.51	8	98.51	1
19-0350	355	2759	201074	11	1	6	98.30	6	98.30	1
20-0420	237	2249	87077	1	1	10	97.83	10	97.83	1
21-0500	326	1263	113594	1	1	4	93.18	4	93.18	1
22-1750	1799	15125	813037	43	2	7	98.69	7	98.69	7
23-1800	2712	30625	455735	8	4	9	99.28	9	99.28	5
24-2000	1849	12452	567396	7	2	7	98.53	7	98.53	2
25-2230	4977	6997	610084	10	1	3	96.91	3	96.91	1
26-2300	1269	11492	635123	7	1	7	98.04	7	98.04	2
27-2550	1576	4655	588188	14	1	5	84.06	5	84.04	1
28-2800	1523	10523	2087947	93	2	3	96.95	3	96.96	6
29-2900	15664	26117	1477634	43	3	6	99.57	6	99.57	3
30-3000	3398	29903	1942250	178	5	7	95.84	7	97.42	20
31-0400	285	1359	273538	75	1	3	42.06	5	39.46	1
32-0550	573	4444	448436	42	1	6	98.80	6	98.81	3
33-0650	578	4053	233788	12	1	5	96.42	5	96.46	2
34-0750	607	4016	3298471	13	1	4	99.46	4	99.46	1
35-1500	1379	10344	844907	51	2	6	97.12	6	97.17	7
36-2000	1534	8533	750979	18	1	7	96.07	7	96.07	1
37-2250	2407	20146	1531733	150	4	5	98.84	5	98.85	13
38-2500	3112	29510	1460508	191	6	3	98.79	3	98.86	31
39-2750	2056	10549	1343881	355	4	2	35.25	2	36.11	4
40-3000	3078	25235	1873230	199	5	4	98.54	4	98.57	20

computing time in the last column, expressed in seconds. Hence, we can compare PIC$^-$ with the specific AC3 in terms of filtering quality by first regarding the consistent level then the filtered domain size.

The qualitative improvement made by adding PIC$^-$ to the specific AC3 consists, firstly, in a refinement of the consistent level for two instances: 01-0200 (from 3 to 4) and 31-0400 (from 3 to 5). In the latter instance, this avoids the resolution method having to search CSP(3) and CSP(4). Secondly, it deletes more values for other instances in a time that remains reasonable, below 31 seconds.

3. Resolution with C$\mathcal{N}$-Tabu

C$\mathcal{N}$-Tabu (for *Tabu Search* on a Consistent $\mathcal{N}$eighbourhood) is an interesting hybrid method which includes some of the arc-consistent mechanisms in its *Tabu Search* (as defined in (Glover and Laguna, 1997)). This method has been applied to some industrial problems adjacent to the FAPP (Vasquez, 2002): the Daily Photographs Satellite Problem (DPSP) (Vasquez and Hao, 2001b) and the Antenna Positioning Problem (APP) (Vasquez and Hao, 2001a), which is the upstream problem of the FAPP. For better understanding, we will first present the general methodology of a *Tabu Search*, then the principle of our hybrid method. The last part explains how our *Tabu Search* is used through the levels corresponding to each CSP, in both directions, downward and upward.

3.1. Tabu search methodology

A *Tabu Search*, TS, is a meta-heuristic designed for tackling hard combinatorial optimisation problems. By contrast with random approaches, TS is based on the belief that an intelligent search should include more systematic forms of guidance based on adaptive memory and learning. TS can be described as a form of a neighbourhood search with a set of critical and complementary components.

For a given optimisation instance (S, f) characterised by a search space S and an objective function f, a neighbourhood $\mathcal{N}$ is introduced. It associates for each s in S, a non-empty subset $\mathcal{N}(s)$ of S. A typical TS algorithm begins with an initial configuration s in S, then repeatedly visits a series of the best local configurations following the neighbourhood function. At each iteration, one of the best neighbours $s' \in S$ is selected to become the current configuration, even if s' does not improve the current one in terms of cost function. To avoid the problem of cycles occurring and allow the search to go beyond local optima, tabu list is introduced. This adds a short time memory component to the method. A tabu list maintains a selective history H (short time memory), composed of previously encountered configurations or, more generally, pertinent attributes of such configurations. A simple TS strategy consists in preventing configurations of H from being considered on the k next iterations, called the *tabu tenure*. A tabu tenure can vary for different attributes, and is generally problem dependent. At each iteration, TS looks for the best neighbour from this dynamically modified neighbourhood $\mathcal{N}(H, s)$, instead of $\mathcal{N}(s)$ itself. Such a strategy prevents the search from being trapped in short term cycling and makes the search more rigorous. When attributes of configurations, instead of configurations themselves, are recorded in a tabu list, some unvisited, yet interesting configurations may be prevented from being considered. Aspiration criteria may be used

to overcome this problem. A widely used aspiration criterion consists in removing a tabu status from a move when it leads to a configuration better than the best one obtained so far. Since TS uses an aggressive search strategy to exploit its neighbourhood, it is crucial to have special data structures and techniques which allow rapid updating of move evaluations, and reduce the effort of finding best moves. A *Tabu Search* can therefore be described by specifying its main elements: the configuration representation, the cost function to evaluate the configurations, the neighbourhood function, the tabu list and its tabu tenure, and finally the aspiration criterion.

3.2. The C$\mathcal{N}$-Tabu method

For better understanding of the method description, we must differentiate between two uses of the term *neighbouring*. The first one concerns the *neighbouring variables* in the constraint network, which are the variables linked by a constraint to the given variable; whereas a *neighbouring configuration* in an approximate method is used to denote a new configuration obtained from the current one by a move which changes one or more variable affectations.

Now, we will see the specificities of our *Tabu Search*. The first and most important point to emphasize is that C$\mathcal{N}$-Tabu works on *consistent partial configurations*, rather than on inconsistent complete configurations, like all the other local search methods. Hence, the cost function used is merely the affected variable number in the configuration: one configuration is better than another if it has more instantiated variables.

In order to visit the combinatorial space search, C$\mathcal{N}$-Tabu jumps from a configuration to a neighbouring one by making moves. A move consists in affecting a not yet instantiated variable, then repairing the possible conflicting affectations. More precisely, each move is made in two steps: first, we assign a pair (f_i, p_i) to the chosen candidate path x_i, then we propagate this affectation to its neighbours x_j in the constraint network and, if necessary, we de-instantiate the conflicting neighbouring values, using local considerations respecting $\mathrm{IC}(ij) \cup \mathrm{ECC}_k(ij)$. This can be done efficiently using the incremental computing principle as in (Fleurent and Ferland, 1996), on specific data structures, allowing variable domains to be dynamically reduced. More precisely, after each move, we update the domains of x_j, neighbours of the variable x_i which has already been instantiated, by considering their inconsistency degrees regarding the current partial configuration. Hence, this mechanism can be viewed as a dynamic filtering operation. As the complexity of the updating step is directly linked to the graph constraint degree, the mechanism seems to be appropriately sized only for weak arity networks. Moreover, a tabu list is needed to prevent cycling, which notably occurs when we attempt to instantiate the last uninstantiated variables in the current partial configuration. Indeed, all the domain values (f_j, p_j) likely to de-instantiate the variable x_i affected by the move are classified tabu during some iterations: the tabu tenure is proportional to the number of times this resource has been affected. However, the aspiration criterion enables selection of a tabu neighbour if and only if it improves the cost function, i.e., it has more affected variables than the current one. Finally, the last two important elements

in a *Tabu Search* are the *intensification* and the *diversification* phases. Given that our study focuses on minimising the relaxation level, we need to solve the different CSP(k) as quickly as possible. The intensification phase is therefore not needed. The diversification phase makes it possible to escape from attractive zones of the search space. For this purpose, we introduce penalties during the search phase. More precisely, each time a partial configuration s cannot be extended (i.e., the configuration s has more instantiated variables than all its neighbours), we add a penalty to all pairs $(x_i, (f_i, p_i))$ of the affected variables which have an unassigned neighbour in the constraint network. These assigned variables belong to a *nogood*, emphasized by adding penalties. This penalty value is then included in the move heuristic during the diversification phase. Indeed, the diversification phase is a tabu search in which candidate selection depends on the "*nogood*" values.

4. Downward and upward methods

The presented filtering algorithm provides the last consistent level. However, this level may be lower than the feasible one, i.e., the minimal level from which a solution exists. If they are the same (i.e., the search method found a solution at the last consistent level returned by the filtering), we are sure that this level is the optimal one, considering the first point of the objective function (i.e., minimize the feasible level).

This allows to envisage two ways to tackle the FAPP: the first one starts from the maximal level, then decreases the level when it has found a solution. The second one, in opposite, starts from the last consistent level, and try to find a solution. If not, it increases the level since it found a feasible level.

In the next, we will present the two approaches, *downward* and *upward*, by using the C$\mathcal{N}$-Tabu algorithm to solve each CSP(k), before giving some comparison results.

4.1. Downward approach

Downward is the most intuitive approach, knowing that a local search can say if it has found a solution but never say if no solution exists. Starting from level $k = 11$ where any ECCs was considered, by a consistent partial configuration provided by a *Greedy* search, *downward* attempts to decrease the feasible level to the consistent one. So, the main loop alternates search and diversification phases, in order to solve each CSP(k) until the last consistent level and decreases k if a solution is found.

Algorithm 1 describes the *downward* method, where S is the search space, and f the objective function. For more clarity, we detail the notations and functions involved:

- s^* represents the optimal configuration;
- s^- represents the partial configuration from level $k - 1$, obtained from the solution at level k by de-instantiating the conflicting variables;
- n is the total number of variables;
- k is the relaxation level, where the algorithm solves the CSP(k);

```
Algorithm 1: DOWNWARD(S, f)
begin
    k ← 11
    s ← Greedy(k)
    CN-Tabu(k, s)
    while (not stop-criterion) do
        if (|s| = n) then
            k ← k - 1
            get-filtering-domains(k)
            CN-Tabu(k, s⁻)
        else
            if (|s*| ≥ |s|) then
                CN-Tabu(k, s*)
            else
                CN-Tabu-diversification(k, s)
                if (|s| = n) then
                    k ← k - 1
                    get-filtering-domains(k)
                    CN-Tabu(k, s⁻)
                else
                    if (|s*| ≥ |s|) then
                        CN-Tabu-diversification(k, s*)
                    else
                        change-parameters()
                        CN-Tabu(k, s*)
end
```

Algorithm 1.

- *get-filtering-domains*(k): computes, for each variable domain, the consistent values at level k. Indeed, the domains are thus dynamic;
- *change-parameters*(): updates the execution parameters of C$\mathcal{N}$-Tabu (seed, maximal iteration number, etc.);
- *stop-criterion*: means that a solution is found at the consistent level (k-const), or once the authorised time has elapsed.

4.2. Upward approach

By contrast with *downward*, *upward* starts from the consistent level and climbs through the levels. It attempts to solve the different CSP(k) between the consistent and the feasible levels. More precisely, while no solution is found at the current level, k is increased and the new CSP(k) resolution starts from the best partial configuration produced by the previous step. Once a solution is reached, the feasible level becomes k. The aim is

Table 2
Upward versus downward.

Instance	k	Downward		Upward		Instance	k	Downward		Upward	
		$T\%$	time	$T\%$	time			$T\%$	time	$T\%$	time
01-0200	$\underline{4}$	100	0.75	100	0.44	21-0500	$\underline{4}$	100	0.01	100	<0.001
02-0250	$\underline{2}$	100	3.55	100	4.12	22-1750	$\underline{7}$	100	92.03	100	0.92
03-0300	$\underline{7}$	100	13.68	100	16.05	23-1800	$\underline{9}$	77	7.87	100	1.75
04-0300	$\underline{1}$	98	25.48	100	10.49	24-2000	$\underline{7}$	100	6.52	100	0.05
05-0350	11	100	<0.001	100	0.88	25-2230	$\underline{3}$	100	6.77	100	0.39
06-0500	$\underline{5}$	97	204.44	99	35.05	26-2300	$\underline{7}$	100	8.56	100	1.01
07-0600	$\underline{9}$	83	3.91	100	3.21	27-2550	$\underline{5}$	100	23.92	100	2.55
08-0700	$\underline{5}$	98	39.33	99	7.1	28-2800	$\underline{3}$	100	33.64	100	1.2
09-0800	$\underline{3}$	100	14.78	100	7.48	29-2900	$\underline{6}$	100	17.31	100	2.03
10-0900	$\underline{6}$	100	869.42	98	7.64	30-3000	$\underline{7}$	100	319.55	100	5.99
11-1000	$\underline{8}$	95	441.31	98	22.66	31-0400	$\underline{5}$	65	791.86	18	809.33
12-1500	$\underline{2}$	0	–	36	1371.53	32-0550	$\underline{6}$	99	68.86	100	0.13
13-2000	$\underline{3}$	0	–	3	2242	33-0650	$\underline{5}$	79	112.47	100	<0.001
14-2500	$\underline{4}$	1	3366	2	2542	34-0750	$\underline{4}$	94	10.04	99	5.53
15-3000	$\underline{5}$	77	2332.87	83	1748.52	35-1500	$\underline{6}$	48	108.02	100	2.24
16-0260	$\underline{11}$	100	<0.001	100	<0.001	36-2000	$\underline{7}$	87	167.95	97	10.39
17-0300	$\underline{4}$	100	<0.001	100	<0.001	37-2250	$\underline{5}$	14	834.5	100	1.66
18-0350	$\underline{8}$	100	<0.001	100	<0.001	38-2500	$\underline{3}$	42	1168.67	100	4.96
19-0350	$\underline{6}$	100	2	100	<0.001	39-2750	3	0	–	0	–
20-0420	$\underline{10}$	100	<0.001	100	<0.001	40-3000	$\underline{4}$	64	722.81	98	15.8

thus to reduce the feasible level to the consistent one. Algorithm 2 describes the *upward* method as an "intelligent" algorithm, which reuses information and configurations from the previous resolution. However, this approach is more risked. Indeed, *downward* approach guarantees to go down from a level only if a solution was found at the current level, whereas it may be possible that *upward* searches at several levels without never finding a solution.

Algorithm 2 describes *upward* method. The notations are the same as *downward* algorithm, on which we add:

- *k-const*: the relaxation level returned by the filtering process, it means that no solution can be found at (k-*const* $-$ 1) level, at less one variable domain has been empty by constraint propagations.
- *k-real*: the level at which C$\mathcal{N}$-Tabu has found a solution. So, the aim is to reduce the *k-real* level to the k-const one.

4.3. Result comparisons

Tables 2 present the results obtained by 100 runs of one hour on each instance, with the initial seed varying from 0 to 99. They compare the *downward* (columns 3 to 4) and *upward* (columns 5 and 6) approaches. Column 1 is the instance name and column 2 the obtained level, it is underlined if it is the optimal level. For each method (*downward*,

```
Algorithm 2: UPWARD(S, f)
begin
    k-real ← 11
    get-filtering-domains(k-const)
    s ← Greedy(k-const)
    CN-Tabu(k-const, s)
    while (k-real ≠ k-const or not stop-criterion) do
        if (|s| = n) then
            k-real ← k
            if (k-real = k-const) then End
            else
                get-filtering-domains(k-const)
                CN-Tabu(k-const, s⁻)
        else
            if (|s*| ≥ |s|) then
                CN-Tabu(k, s*)
            else
                CN-Tabu-diversification(k, s)
                if (|s| = n) then
                    k-real ← k
                    if (k-real = k-const) then End
                    else
                        get-filtering-domains(k-const)
                        CN-Tabu(k-const, s⁻)
                else
                    if (|s*| ≥ |s|) then
                        CN-Tabu-diversification(k, s*)
                    else
                        if (k = k-real) then
                            change-parameters()
                            get-filtering-domains(k-const)
                            CN-Tabu(k-const, s*)
                        else
                            get-filtering-domains(k + 1)
                            CN-Tabu(k + 1, s*)
end
```

Algorithm 2.

and *upward*), we give the success rate in the $T\%$ column, as well as the average time in seconds required to reach the optimal level, in the *time* column.

By comparing the number of times each method found the optimal level, we see that *upward* was 100% reliable on 27 instances, whereas *downward* only worked on 20 instances. Even if we consider that a method can be said to be reliable if it has a ratio

$\geqslant$75%, *downward* is reliable on 31 instances, as compared with 35 for *upward*. These results show that the *upward* approach is more stable than the *downward* one (i.e., it is less dependent on the tuning parameters, especially the random seed).

Furthermore, the *upward* approach consumes less time than *downward*. Indeed, *upward* is faster than *downward* on 29 instances, whereas *downward* is faster on only 4 instances. For most of the benchmark instances, the times are approximately comparable. However, some need to be focused, notably instance 22-1750, where *upward* is a hundred times faster than *downward*, or instance 32-0550, where *upward* is more than 500 times faster. Hence, much more time is required in order to improve the other two components of the objective function (i.e., minimise the violated constraint number at level $k-1$, and the sum at level less than $k-1$).

The risks taken by *upward* in its resolution, changing the relaxation level without having necessarily found a solution on the current level, allows this method to produce better results, in terms of quality (or stability) as well as in execution speed, than the *downward* one.

Another aspect to point out concerns the hard instances 12-1500, 13-2000 and 14-2500. The *upward* approach found the optimal level 36 times for 12-1500, and 3 times for 13-2000 as compared with 0 times by the *downward* method. For instance 14-2500, *upward* found the optimal level twice, as opposed to once by the *downward* approach. These instances are said to be *hard* because they have many variables, many constraints, and the filtering ratio is less than 66%. However, there are two things wrong, concerning the instances:

- 31-0400: the *downward* approach obtains better results than the *upward* approach. This instance has only 400 variables and all (except 6 for *upward*) the founded solutions are at the level 5 or at level the 11.
- 39-2750: no method found a level better than 11, whereas the known optimum is 3; except for two occasions where *upward* found solutions at level 8.

Table 3 details the results obtained on the *hard* instances, in order to better understand the real difficulties met by a local search on these instances. The first two columns give the instance name and the name of the resolution approach, the next ten show the relaxation level. The optimal k is underlined if it has been found, otherwise an * is inserted.

Once again, we note the effectiveness of the *upward* approach, notably on instance 14-2500, where the k found is mainly between 5 and 6, while it can reach 11 by *downward*.

5. Related work

The Frequency Assignment Problem Framework has been widely studied, notably in the context of the European Project EUCLID CALMA (Combinatorial Algorithms for Military Applications). In 1995, the CELAR (Centre d'ELectronique de l'ARmement,

Table 3
More details on hard instances.

Instance	Method	2	3	4	5	6	7	8	9	10	11
12-1500	downward	*	2	16	15	34	33				
	upward	36	9	45	6	1	3				
13-2000	downward		*	11	47	40	2				
	upward		3	57	27	13					
14-2500	downward			1	33	49	4		1	9	3
	upward			2	46	51		1			
31-0400	downward				65						35
	upward				18	6					76
39-2750	downward		*								100
	upward		*					2			98

France) proposed a set of benchmarks for the frequency affectation problem called the RLFAP (Radio Link Frequency Assignment Problem) coming from real networks with simplified data. From the wide range of literature in the field (Koster, 1999; Aardal et al., 2002; Hao, Dorne, and Galinier, 1998; Castellino, Hurley, and Stephens, 1996; Voudouris and Tsang, 1998), we would like to cite (Aardal et al., 2001), which provides a very clear survey of the different frequency assignment problems and their resolution methods.

The model studied in this paper can be viewed as a follow-up to that of the CALMA project. Indeed, the problem is extended to take into consideration the polarisation and the relaxation level on the electromagnetic compatibility constraints. It was the subject of the ROADEF'01 challenge, which took place at FRANCORO III, in the city of Québec (Canada) in May 2001. It has opposed six finalist methods. We present in table 4 the results they obtained. During the competition, only one run was allowed and the computing time was limited to one hour on a Pentium III, 500 MHz, 128 MB. Table 4 details the hierarchical objective function, by giving first the relaxation level k, then the sum of all the unsatisfied ECC_{k-1}, and finally the sum of all the unsatisfied ECC_i, where i varies from 0 to $k-2$.

The first approach, developed by Bisaillon's team (Galinier et al., 2002) and referred to as TS–VN, is a local search based on *Tabu Search* on a *variable neighbourhood*. This method successively treats three different problems:

- finding a solution at the level k (k-feasible);
- finding a $k+1$-feasible solution which as far as possible satisfies constraints at level k;
- finding a $k+1$-feasible solution and minimising the objective function.

The algorithm MH + CP, developed by Caseau, combines constraint propagation with meta-heuristics, increasing the level. More precisely, at each level, it "shaves" with strong consistency. If the constraint network is consistent, a *Large Neighbourhood Search* (Shaw, 1998) returns a configuration. If it is a nonfeasible solution, then a *Limited Discrepancy Search* (Harvey and Ginsberg, 1995) tries to obtain a feasible one. If not, it increases the level.

Table 4
Comparison with others methods.

FAPP	TS–VN			MH + CP			LS–CC			LNS + CP			Tabu			C$\mathcal{N}$-Tabu		
	k	V1	SV2	k	V1	SV2	k	V1	SV2	k	V1	SV2	k	V1	SV2	k	V1	SV2
01-0200	4	4	56	4	6	279	4	14	165	4	5	210	5	1	281	4	14	233
02-0250	2	7	86	2	18	248	2	21	160	11	1	435	11	1	1274	2	20	195
03-0300	7	10	341	7	27	1076	7	16	420	11	1	1211	7	13	589	7	32	892
04-0300	1	31	0	1	164	0	3	9	224	2	1	282	7	1	3678	1	184	0
05-0350	11	1	372	11	892	12364	11	1	1467	11	97	3459	11	7	2284	11	364	5694
06-0500	5	12	246	5	53	1029	7	15	879	6	1	1086	7	15	1210	5	31	811
07-0600	9	22	714	9	132	4419	10	28	3070	12	–	–	9	33	1585	9	106	3375
08-0700	5	16	266	5	53	1359	5	37	691	11	3	2144	5	26	625	5	73	1225
09-0800	3	28	195	3	63	937	4	24	573	4	2	999	10	1	3678	3	104	846
10-0900	6	18	475	6	82	2365	6	39	1146	11	4	3661	8	5	2871	6	103	2003
11-1000	8	8	1015	8	119	5206	9	30	3736	11	8	6146	10	1	5108	8	119	4191
12-1500	3	83	1698	7	180	6538	11	17	2634	11	647	13797	9	70	7682	2	62	1310
13-2000	3	49	2003	7	229	7503	11	59	6164	11	671	15145	10	13	9651	5	132	3645
14-2500	4	35	3485	8	18	10661	11	3	5574	11	1209	24751	10	101	15718	5	217	5045
15-3000	5	15	1569	7	333	9988	11	46	9523	11	1060	22898	10	61	14010	5	192	4727
16-0260	11	5	56	11	572	5779	11	67	913	11	590	5968	11	5	57	11	514	5189
17-0300	4	4	34	4	4	36	4	4	35	4	4	36	4	4	34	4	4	36
18-0350	8	4	55	8	4	55	8	4	57	8	4	57	8	4	55	8	4	59
19-0350	6	2	51	6	3	79	6	2	53	6	2	60	6	2	51	6	3	70
20-0420	10	5	97	10	6	145	10	5	99	10	5	106	10	5	97	10	7	142
21-0500	4	2	10	4	2	12	4	2	11	4	2	12	4	2	10	4	2	12
22-1750	7	15	187	7	16	356	7	16	194	7	15	292	7	15	187	7	25	503
23-1800	9	16	187	9	17	197	9	16	189	12	–	–	9	16	187	9	17	197
24-2000	7	6	71	7	7	90	7	7	79	7	6	77	7	6	71	7	9	91
25-2230	3	7	32	3	7	33	3	7	33	3	7	34	3	7	32	3	7	33
26-2300	7	9	74	7	10	81	7	9	74	7	9	75	7	9	74	7	10	86
27-2550	11	4	64	5	7	46	5	8	37	5	4	22	5	4	20	5	11	54
28-2800	3	13	32	3	32	129	3	25	58	3	14	72	3	13	32	3	42	142
29-2900	6	25	239	6	28	351	6	25	212	12	–	–	6	25	212	6	25	310
30-3000	11	1166	12029	7	17	602	7	16	190	12	–	–	7	13	148	7	48	1045
31-0400	5	4	1180	5	161	2131	5	34	1151	5	63	1845	5	16	1400	5	117	1896
32-0550	10	52	1739	6	16	388	6	5	71	11	116	2057	11	25	2166	6	10	235
33-0650	5	7	66	5	16	332	5	7	77	5	7	181	11	5	1310	5	10	235
34-0750	4	2	46	4	35	767	4	6	213	4	3	433	10	1	1701	4	22	565
35-1500	7	3	1280	6	74	1919	6	16	431	11	324	5967	11	24	5870	6	62	1375
36-2000	7	99	2153	9	3	2478	8	25	970	7	19	1451	11	16	4652	7	63	1643
37-2250	11	3	12229	5	56	1745	8	13	975	11	4703	52406	11	14	10353	5	51	1288
38-2500	11	79	14058	3	39	572	3	14	174	12	–	–	11	53	13355	9	125	6717
39-2750	3	356	2844	3	2567	10470	3	747	4603	3	2211	9498	11	36	13267	11	394	740473
40-3000	11	39	16755	4	77	1562	8	20	1261	12	–	–	11	867	13684	4	64	1252

The third method, LS–CC, developed by Gavranovic is a typical *Local Search* guided by the *constraint cost*. At each level, it builds frequency trees, ignoring the polarisation constraints. Then it tries to optimise the polarisation allocation.

In a similar way, the classical *Tabu Search* (referred to as *Tabu* in table 4), implemented by Schindl's team decreases the level, and at each level optimises a frequency allocation without any polarisation, then assigns polarisation values.

Michelon junior team solves the FAPP by using a meta-heuristic based on a *Large Neighbourhood Search* and on constraint propagation (LNS + CP). They relax some constraints to obtain a maximal cover tree. Indeed, their algorithm works in three phases: first it computes a lower bound of k, then it searches for a k-feasible solution, and finally it improves this solution with respect to the last two points of the objective function.

Finally, the last column gives the results obtained by the C$\mathcal{N}$-Tabu developed for the challenge. It follows the *downward* approach, using a standard AC procedure.

Briefly to analyse table 4, we only focus on the optimal level k^*. The first method (TS–VN) reaches this level in 32 instances, the second (MH + CP) in 35 instances, and the third (LS–CC) in 28 instances. Both LNS + CP and *Tabu* reach k^* only in 19 instances. By achieving optimality 36 times, C$\mathcal{N}$-Tabu obtains the best results in the competition. Moreover, the improvements presented in this paper are significant. Indeed, *downward* finds k^* for 37 instances and *upward* fails only in one instance (see table 2). However, great care is needed, because the running conditions are not the same: although the resolution time was fixed to one hour, CPU speeds are different, and we consider the best results out of 100 runs.

More details on this challenge, the benchmarks, and all the results can be found on the web site: `http://www.prism.uvsq.fr/~vdc/ROADEF/CHALLENGES/2001/challenge2001_en.html`

6. Conclusion and perspectives

In this paper, we have presented an original approach to the resolution of the FAPP. It combines a specific filtering algorithm with the hybrid C$\mathcal{N}$-Tabu search.

The filtering algorithm, based on the FAPP constraint specificities and improved by considering the *path inverse consistency* on 3-cliques, refines the consistent level search. From a theoretical point of view, the PIC algorithm for the 3-cliques on ECCs, could be generalised to the *k-inverse consistency*. The k-inverse consistent domain of D_i becomes the intersection between D_i and the $k!$ intervals defined by the $k!$ orders. The complexity remains linear within the domain size.

C$\mathcal{N}$-Tabu hybridises a *Tabu Search* with arc-consistent mechanisms, by considering the consistent neighbourhood. Efficient at solving CSPs, it is the kernel of the two hierarchical resolution strategies *downward* and *upward*. Emphasised by the experimental results, the speed and stability of *upward* confirm the effectiveness of working on partial but consistent configurations. Moreover, they enable the other two components of the objective function to be tackled. To achieve this, we envisage to define equality constraints of distances between frequencies as meta-variables because of their hardness. One last point that should be highlighted concerns instance 39-2750, for which a more precise analysis is needed to understand its constraint network particularity, in order to improve the C$\mathcal{N}$-Tabu.

Acknowledgments

Special thanks to Van-Dat Cung (PR*i*SM-UVSQ), Thierry Defaix (CELAR-DGA) and Maurice Diamantini (ENSTA) for allowing us to use their FAPP definition. Thanks also to Eric Nabor from THALES-COMMUNICATIONS for helpful discussions.

References

Aardal, K., C. Hurkens, J. Lenstra, and S. Tiourine. (2002). "Algorithms for Radio Link Frequency Assignment: The CALMA Project." *Annals of Operations Research* 50(6), 968–980.

Aardal, K., C. van Hoesel, A. Koster, C. Mannino, and A. Sassano. (2001). "Models and Solution Techniques for the Frequency Assignment Problem." Technical Report (ZIB-Report 01-40), Konrad-Zuse-Zentrum für Informationstechnik Berlin, Maastricht University.

Bessière, C., E.C. Freuder, and J. Régin. (1995). "Using Inference to Reduce Arc Consistency Computation." In *International Joint Conference on Artificial Intelligence*, Montreal, Canada, pp. 592–598.

Castellino, D., S. Hurley, and N. Stephens. (1996). "A Tabu Search Algorithm for Frequency Assignment." *Annals of Operations Research* 63, 301–319.

Debruyne, R. and C. Bessière. (1997). "Some Practicable Filtering Techniques for the Constraint Satisfaction Problem." In *Proceedings of the International Joint Conference on Artificial Intelligence*, Nagoya, Japan, pp. 412–417.

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

Freuder, E. and C. Elfe. (1996). "Neighborhood Inverse Consistency Preprocessing." *Proceedings of 13th American Association for Artificial Intelligence*, Portland, Oregon, pp. 202–208.

Galinier, P., S. Bisaillon, M. Gendreau, and P. Soriano. (2002). "Solving the Frequency Assignment Problem with Polarization by Local Search and Tabu." In *6th Triennal Conference of the International Federation of Operational Research Societies (IFORS'02)*, University of Edinburgh, UK.

Glover, F. and M. Laguna. (1997). *Tabu Search*. Dordrecht: Kluwer Academic.

Hao, J., R. Dorne, and P. Galinier. (1998). "Tabu Search for Frequency Assignment in Mobile Radio Networks." *Journal of Heuristics* 4(1), 47–62.

Harvey, W. and M. Ginsberg. (1995). "Limited Discrepancy Search." In *Proceeding of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pp. 607–613.

Hertz, A., D. Schindl, and N. Zufferey. (2001). "A Tabu Search for the Frequency Assignment Problem with Polarization." In *FRANCORO III*, Quebec, Canada.

Koster, A. (1999). "Frequency Assignment – Models and Algorithms." Ph.D. Thesis, Universiteit Maastricht, The Netherlands.

Mackworth, A. (1977). "Consistency in Networks of Relations." *Artificial Intelligence* 8(1), 99–118.

Shaw, P. (1998). "Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems." In *Principle and Pratice of Constraint Programming, CP'98.*

Vasquez, M. (2002). "Arc-Consistency and Tabu Search for the Frequency Assignment Problem with Polarization." *In 4th International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimisation Problems(CP-AI-OR'02)*, Le Croisic, France, pp. 359–372.

Vasquez, M. and J. Hao. (2001a). "A Heuristic Approach for Antenna Positioning in Cellular Network." *Journal of Heuristic* 7, 443–472.

Vasquez, M. and J. Hao. (2001b). "A "Logic-Constrained" Knapsack Formulation and a Tabu Algorithm for the Daily Photograph Scheduling of an Earth Observation Satellite." *Computational Optimization and Applications* 20, 137–157.

Voudouris, C. and E. Tsang. (1998). "Solving the Radio Link Frequency Assignment Problem Using Guided Local Search." In *Proceedings of the NATO Symposium on Radio Length Frequency Assignment, Sharing and Conservation Systems (Aerospace)*, Aalborg, Denmark, Paper 14.

Waltz, D. (1975). "Understanding Line Drawing of Scenes with Shadows." In P.H. Winston (ed.), *The Psychology of Computer Vision*, pp. 19–91. New York: McGraw-Hill. First parution in Technical Report AI271, MIT, Cambridge, MA (1972).