RAIRO Operations Research
RAIRO Oper. Res. **37** (2003) 311-323
DOI: 10.1051/ro:2004004

CONSISTENCY CHECKING WITHIN LOCAL SEARCH APPLIED TO THE FREQUENCY ASSIGNMENT WITH POLARIZATION PROBLEM

Michel Vasquez[1], Audrey Dupont[1] and Djamal Habet[1]

Abstract. We present a hybrid approach for the Frequency Assignment Problem with Polarization. This problem, viewed as Max-CSP, is treated as a sequence of decision problems, *CSP* like. The proposed approach combines the Arc-Consistency techniques with a performed *Tabu Search* heuristic. The resulting algorithm gives some high quality solutions and has proved its robustness on instances with approximately a thousand variables and nearly ten thousand constraints.

Keywords. Filtering techniques, consistency checking, *Tabu Search*.

1. Introduction

The ever-increasing demand for communication, coupled with the limited number available spectra, have made frequency assignment more and more difficult to accomplish effectively. Optimization of this process has therefore become a major issue for network administration and deployment, both civil and military. The Frequency Assignment Problem with Polarization (*FAPP*) can be formalized as a Max-CSP which is known to be *NP-hard*.

Because the data problem to be optimized was very large, we decided to adopt a local search method. From among all the existing algorithms, we chose *Tabu Search*, introduced by Glover in [8]. The specific structure of the *FAPP* constraint network led us to apply local filtering techniques. Consequently, an arc-consistency procedure, AC, was embedded in the *Tabu Search* framework to reduce the search

[1] Centre LGI2P, École des mines d'Alès, site EERIE, 30035 Nîmes Cedex 1, France;
e-mail: vasquez,dupont,habet@site-eerie.ema.fr

space. Hence, the consistency checking concept occurred twice (during the resolution): firstly in the filtering pre-processing, and secondly in the kernel of the neighborhood design.

After presenting the physical and formal definitions of the *FAPP* problem in Section 2, we describe in Section 3 our general approach to solve it. Finally, a large number of experimental results are discussed and compared in Section 4, in order to highlight the advantages obtained by combining a constraint programming tool with the local search heuristic.

2. Problem definition

2.1. Physical description

The *FAPP* consists in finding an optimal frequency allocation in hertzian telecommunication networks. The network is composed of a set of sites in which the transmission devices (antennae connected to emitters or receptors) are located. A hertzian connection joins two geographic sites by one or more paths. A path is a uni-directional radio-electric bond, established between antennae at distinct sites, which has a given frequency and polarization.

A frequency resource is therefore a (frequency, polarization) pair in which the components are respectively associated to the carrying frequency of the transmitted signal, and the wave polarization. The polarization is simply either positive or negative. Accordingly, we define the path domain as a set of available resources. This set contains the frequency domain F_i, and the polarization information P_i, which may include a required polarization.

A frequency allocation consists in assigning a (f_i, p_i) pair for each path i which satisfies certain radio-electric compatibility constraints (1, 2, and 3), and minimal distance constraints to avoid interference (4):

1. frequencies equality or not across paths i and j: $f_i = f_j$ or $f_i \neq f_j$;
2. distance between frequencies: $|f_i - f_j| = \varepsilon_{ij}$ or $|f_i - f_j| \neq \varepsilon_{ij}$;
3. polarization equality or not across paths: $p_i = p_j$ or $p_i \neq p_j$;
4. minimal distance between frequencies: $|f_i - f_j| \geq \begin{cases} \gamma_{ij} & \text{if} \quad p_i = p_j \\ \delta_{ij} & \text{if} \quad p_i \neq p_j. \end{cases}$

In the constraints (4), the required distance between frequencies depends on their polarizations: the distance is obviously smaller if the polarizations are different ($\gamma_{ij} \geq \delta_{ij}$).

A feasible solution is an allocation for each path that satisfies the full set of constraints. Unfortunately, most problems do not have any such feasible solution, because the domains are too restrictive or the requirements are too numerous. The operator must therefore search for a "good quality" solution in terms of interference, rather than a feasible one. For this purpose, two constraint classes are introduced:

- CI: strong or imperative constraints (1, 2, and 3 above);
- CEM: constraints of type 4 where progressive relaxation is authorized and controlled by 11 relaxation levels. Level 0 corresponds to no relaxation. Increasing from level k to $k+1$ involves relaxation of some or all the frequency distances defining each constraint, the maximum level being 10:

$$|f_i - f_j| \geq \begin{cases} \gamma_{ij}^0 \geq \gamma_{ij}^1 \geq ... \geq \gamma_{ij}^k \geq ... \geq \gamma_{ij}^{10} & \text{if } p_i = p_j \\ \delta_{ij}^0 \geq \delta_{ij}^1 \geq ... \geq \delta_{ij}^k \geq ... \geq \delta_{ij}^{10} & \text{if } p_i \neq p_j. \end{cases}$$

In this context, a feasible solution at level k is an allocation of each path satisfying all the strong constraints CI and all the CEM constraints at level k, noted CEM_k. Such a problem is said to be k-feasible. Assuming that a good quality solution minimizes interference, and that the smaller k is the fewer CEM constraints are relaxed, the hierarchical objective function is dependent on k: first minimizing k, secondly minimizing the number of unsatisfied CEM_{k-1} constraints, and finally minimizing the number of unsatisfied CEM constraints at the levels below $k-1$.

2.2. Formal definition

From this physical description, modelling the *FAPP* as a Maximal Constraint Satisfaction Problem (Max-CSP), consists in defining the constraint network $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, and the formal criterion to minimize f:

- associating a *variable* x_i to each path: $\mathcal{X} = \{p_i, i = 1, ..., n\}$;
- associating to x_i a *domain* $D_i = F_i \times P_i$, where F_i is the set of the allowed frequencies for the path x_i and P_i is one of the following sets {-1}, {1}, or {-1, 1}, $\mathcal{D} = \bigcup_{i=1,\ldots,n} D_i$;
- adding several imperative constraints CI between two paths x_i and x_j: $f_i = f_j$ or $f_i \neq f_j$, $|f_i - f_j| = \varepsilon_{ij}$ or $|f_i - f_j| \neq \varepsilon_{ij}$, $p_i = p_j$ or $p_i \neq p_j$;
- adding CEM constraints where progressive relaxation is authorized:

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

  where the index k, increasing from 0 to 10, indicates the relaxation level of the CEM constraints.
  $\mathcal{C} = \text{CI} \cup \text{CEM}$;
- let $V_{(k)}$ be the set of unsatisfied CEM_k constraints. The hierarchical objective function to be minimized is:

$$f = \left(k, V_{(k-1)}, \sum_{i<k-1} V_{(i)} \right)$$

  where k is such that the problem is k-feasible. The formal criterion f is directly related to the hierarchical optimization problem. We must therefore successively minimize k, $V_{(k-1)}$ then $\Sigma_{i<k-1} V_{(i)}$.

Where necessary, for simplicity, we also define:

- $C_k = \mathrm{CI} \cup \mathrm{CEM}_k$ the whole constraint set at level k;
- $\mathrm{CI}(i,j) \subset \mathrm{CI}$ the imperative constraints involving the paths i and j;
- $\mathrm{CEM}_k(i,j) \subset \mathrm{CEM}_k$ the relaxed constraints involving the paths i and j.

3. General approach for solving the *FAPP*

Among all of the dedicated algorithms to solve frequency assignment problems [1,4,11], only heuristic based approaches can produce good quality solutions in reasonable computing time for realistically sized instances. The described algorithm here is also a meta-heuristic one. Although it may seem unworkable to attempt to use a pre-processing filtering operation on an optimisation problem, our *Tabu Search* procedure is reinforced by a consistency technique that reduces the search space and increases the overall efficiency of the neighborhood exploration. Indeed:

- let n be the path number of a *FAPP* instance;
- let S be the non constrained search space: S includes all the configuration vectors $s = ((f_1, p_1), ..., (f_n, p_n))$ such as $f_i \in F_i$ and $p_i \in P_i$.

The initial size of S, equal to $\prod_1^n |F_i| \times |P_i|$, is huge, even for the smallest *FAPP* instances ($200 \leq n \leq 3000$ and $19 \leq |F_i| \leq 500$). To cope with this difficulty, we restrict the F_i domains by a filtering process.

The two expected advantages of such a reduction in domain size are: firstly to decrease the time complexity of the neighborhood evaluation, and secondly to avoid exploring wrong areas in the search space. That is why we hybridize our *Tabu Search* algorithm with an Arc-Consistent (AC) filtering process.

Another important feature of this approach, with the aim of reducing computing complexity, is that only the first component k of the objective function f is considered within the move heuristic. Consequently, the optimization process is transformed into a sequence of decision problems verifying the existence of a k-feasible configuration, where k decreases from 11 to k^* (best k found). The search for a lower value of k is started only if a $(k+1)$-feasible configuration has been found. Hence, we solve several CSPs rather than one Max-CSP.

To summarize, if AC is the filtering function of the frequency domains, and Tabu the exploring function of the search space using the meta-heuristic *Tabu Search*, we have the general pattern described in Algorithm 1.

Algorithm 1. AC-Tabu

```
begin
    k ← 11
    while AC(k) = True do
        if Tabu(k) = True then
            k ← k* − 1
end
```

This is a very straightforward iterative procedure. When either AC(k) or TABU(k) fails then *AC-Tabu* returns $k^* = k + 1$ as the best k^*-feasible solution found. If AC($k-1$) fails and TABU(k) finds a solution, then k is the *optimal* value. Indeed, AC provides the Lower Bound of k, and TABU the Upper one.

On the other hand, if TABU(k) finds a feasible solution, that solution may be k^*-feasible, with $k^* < k$. This means that TABU can jump more than one k-level.

3.1. Filtering with AC

Arc-consistency is a widely studied topic in constraint programming [2,3,12,13]. AC eliminates variable values with no support, since such values cannot lead to a feasible solution.

As *FAPP* constraints are binary, it is easy to check for every path i ($1 \le i \le n$) and for every value couple (f_i, p_i), if for each neighbor j of i, there is a couple (f_j, p_j) such that CI(i, j) and CEM$_k(i, j)$ are satisfied. This elimination process, $eliminate(F_i, P_i, k)$, is repeated until there is no change in the domain of any variable: in this case *eliminate* returns $False$, otherwise it returns $True$. For more details about such algorithms, please refer to [2, 3, 12]. The main features of the following AC(k) procedure are:

- managing a list of updated variable domains, in order to check, at the next iteration, only the variables for which their domains have changed;
- for each f_i and p_i values, and for each constraint $C(i, j)$, the *eliminate* (F_i, P_i, k) function checks another value in the domain D_i as soon as it has found a pair $f_j \in D_j$ and $p_j \in P_j$ satisfying $C(i, j)$.

Algorithm 2. AC(k)

```
begin
    change ← True
    while change do
        change ← False
        for i ∈ [1, n] do
            if eliminate(F_i, P_i, k) then
                if F_i = ∅ or P_i = ∅ then
                    return False
                change ← True
end
```

3.2. Exploring with Tabu

Contrary to the random local search, where randomness is extensively used, the meta-heuristic *Tabu Search* is based on the belief that an intelligent search should include more systematic forms of guidance based on adaptive memory and learning.

Designing a *good Tabu Search* requires that its main characteristics should be well defined: the effective search space and therefore the definition of the visited configurations, the neighborhood structure, the move heuristic, and the tabu list management.

Algorithm 3. TABU(k)

```
begin
    s ← 0⃗                                   % i.e. any path is allocated: |s| = 0
    s* ← s                                  % the best configuration found so far
    while stop-criterion = False do
        s_max ← 0⃗                           % the best neighbor of s
        for s' ∈ N(s) do
            if Δ(s, s') > 0 ∨ move(s, s') is not tabu then
1               if Δ(s, s') > Δ(s, s_max) then
                    s_max ← s'
        if s_max ≠ 0⃗ then
            s ← s_max                       % move(s, s_max)
            if |s| > |s*| then
                s* ← s
                if |s| = n then
                    return True
            update tabu list
        else
            stop-criterion ← True
    return False
end
```

In our approach, the search space is defined over *partial configurations* expressed by $s = (v_1, v_2, ..., v_{n_i})$, where $v_i = (f_i, p_i)$ and n_i variables are instantiated (with $n_i = |s|$). To conform with the standard representation by *n-ary* vectors, we add a new value u (for *uninstantiated*) to each domain D_i, for indicating that x_i is free. Accordingly, the evaluation criterion of a configuration is the number of instantiated variables in s.

This improvement enabled us to work on a *consistent neighborhood* $\mathcal{N}(s)$. At each level k, the visited configurations in $\mathcal{N}(s)$ respect the CI and the CEM_k constraints. In order to build this original neighborhood, a *move(s,s')*, replacing the current configuration s by $s' \in \mathcal{N}(s)$, is achieved in two steps: an instantiation of a free variable (*i.e.* set to u), followed by consistent reparations, which are simply deinstantiations of the conflicting variables with respect to the C_k constraints. Consequently, the selected move must improve the configuration (*i.e.* increase $|s|$). Hence, the evaluation *heuristic* of moves from s to $s' \in \mathcal{N}(s)$, is simply $\Delta(s, s') = |s'| - |s|$. Note that the Δ evaluation is carried out very quickly using efficient incremental techniques, which are now well known [5, 6, 15, 16].

The tabu list is needed to prevent cycling, which notably occurs when we attempt to instantiate the last free variables. To avoid undoing the recent instantiation (x_i, v_i), we penalize all the conflicting pairs (x_j, v_j) where x_j are the neighbors of x_i in the constraint network, regarding the CI(i, j) and the CEM$_k(i, j)$. In this way, we maintain a table counting the number of times a resource v_l is assigned to a path x_i ($freq[i][l]$++). So, the *tabu tenure* is a dynamic function based on the flip frequencies: $tabu[i][l] = freq[i][l] + iter$, where *iter* is the current iteration number.

The *stop-criterion* is either if a solution is found, or if computing time has elapsed.

With the specifications, we thus present the general TABU algorithm in 3.

3.3. Diversification

The aim of the diversification phase is to enable TABU to escape from attractive zones of the search space. For this purpose, we introduce penalties in the move heuristic. More precisely, each time a configuration s such as $\forall s' \in N(s), |s'| \leq |s|$ is reached, we add a penalty to all the allocated paths having an unallocated neighbor in the constraint graph. This penalty value is then included in the move heuristic during the diversification phase. Hence, the line [1] in Algorithm 3 is replaced by the following code, where $nogood_{\min}$ is the threshold value, and *penalty* is the effective penalty value of the move.

if $(\Delta(s, s') > \Delta(s, s_{\max})) \vee (\Delta(s, s') = \Delta(s, s_{\max}) \wedge penalty < nogood_{\min})$ **then**
  $s_{\max} \leftarrow s'$
  $nogood_{\min} \leftarrow penalty$

4. Numerical experimentation

The algorithms were coded in C programming language. The running tests were carried out on an NT PC station with a Pentium III 600 MHz cpu. The results are presented in four parts. The first describes the reduction of the search space by the filtering step. The second describes the best qualitative results obtained by the hybrid approach *AC-Tabu*. The next section then compares the results produced by our TABU and the hybrid *AC-Tabu* algorithm. This section ends with a comparison between several methods proposed for the ROADEF-2001 challenge. More information about this competition is available on the web site `http://www.prism.uvsq.fr/~vdc/ROADEF/CHALLENGES/2001/`

4.1. Cutting frequency domains

This first table shows some characteristics of the `fapp` instances, their size is given in the second part of their name: up to **3000** variables (paths). These

TABLE 1. Search Space reduction.

fapp	IDS	FDS	$\Delta\%$	k_0	sec.	fapp	IDS	FDS	$\Delta\%$	k_0	sec.
01_0200	26 963	12 712	52.85	2	0	16_0260	47 293	46 622	1.42	10	0
02_0250	36 618	14 759	59.69	1	2	17_0300	64 034	918	98.57	3	0
03_0300	53 536	28 212	47.30	6	1	18_0350	73 016	1089	98.51	7	0
04_0300	61 762	21 962	64.44	0	4	19_0350	201 074	3414	98.30	5	6
05_0350	79 311	54 177	31.69	7	2	20_0420	87 077	1886	97.83	9	0
06_0500	108 024	53 034	50.91	4	6	21_0500	113 594	7745	93.18	3	1
07_0600	109 658	69 952	36.21	8	2	22_1750	813 037	10 656	98.69	6	25
08_0700	134 020	81 933	38.87	4	5	23_1800	455 735	3265	99.28	8	9
09_0800	121 824	52 948	56.54	2	6	24_2000	567 396	8328	98.53	6	4
10_0900	197 665	122 050	38.25	5	8	25_2230	610 084	18 867	96.91	2	6
11_1000	294 634	152 727	48.16	7	15	26_2300	635 123	14 217	97.76	6	4
12_1500	436 967	164 613	62.33	1	70	27_2550	588188	93 768	84.06	4	8
13_2000	320 494	144 873	54.80	2	21	28_2800	2 087 947	63 597	96.95	2	66
14_2500	774 322	320 458	58.61	3	92	29_2900	1 477 634	6435	99.56	5	23
15_3000	515 606	306 127	40.63	4	24	30_3000	1 942 250	80 703	95.84	6	103

problems contain up to **2 087 947** total domain values, and up to **67 898** binary constraints.

Table 1 gives the *Initial* Domain Size ($IDS = \sum_1^n |D_i^0|$) and the *Filtered* one ($FDS = \sum_1^n |D_i^f|$). Columns k_0 indicate the highest unfeasible level encountered by the *AC-Tabu* procedure, and columns *sec.* give the computing time in seconds for the whole iterative filtering process. This table shows that no more than two minutes are required by *AC-Tabu* to reduce the domains of the largest instances of this benchmark.

Apart from the fapp16_0260 problem, these benchmarks can be divided into two subsets. The first one (*from 01 to 15*) contains the instances where the domain size is reduced by nearly half. In the second one (*from 17 to 30*), domains are reduced more than 90%.

4.2. HYBRID APPROACH RESULTS

This part begins with the results obtained by the hybrid approach after 1 computing hour. Only one run (with the 0 random seed) was carried out in this experiment.

For each instance, the Table 2 specifies:

- k, the lowest level where a solution is found. This value is underlined when *AC-Tabu* proves its optimality;
- V_{k-1}, the number of unsatisfied CEM constraints at level $k-1$;
- $\sum V_{k-2}$, the sum of the unsatisfied CEM constraints under the $k-1$ level;
- t_1, the elapsed time in seconds required to reach the best value of k;
- and t_2, the elapsed time required to obtain the best configuration (considering the three components of the objective function).

TABLE 2. *AC-Tabu* results after 1 computing hour.

fapp	k	V_{k-1}	$\sum V_{k-2}$	t_1	t_2	fapp	k	V_{k-1}	$\sum V_{k-2}$	t_1	t_2
01_0200	4	4	203	2	1839	16_0260	11	382	3864	0	2976
02_0250	2	16	173	19	19	17_0300	4	4	36	1	1
03_0300	7	28	835	30	34	18_0350	8	4	55	1	1
04_0300	1	97	0	26	2595	19_0350	6	3	66	16	16
05_0350	11	1	1836	1	3571	20_0420	10	6	133	2	2
06_0500	5	30	764	232	233	21_0500	4	2	12	2	2
07_0600	9	85	3039	107	433	22_1750	7	22	383	76	76
08_0700	5	67	1131	21	35	23_1800	9	16	189	31	31
09_0800	3	71	707	1789	1796	24_2000	7	9	91	17	17
10_0900	6	74	1877	174	372	25_2230	3	7	33	19	19
11_1000	8	105	4278	63	139	26_2300	7	8	75	18	18
12_1500	4	87	2138	2873	2873	27_2550	5	9	46	21	21
13_2000	5	126	3469	1020	1020	28_2800	3	38	125	165	203
14_2500	6	211	6302	1875	2950	29_2900	6	25	280	86	86
15_3000	5	198	4770	2534	2807	30_3000	7	36	798	268	275

TABLE 3. Improved results after 3 hours of computing.

fapp	k	V_{k-1}	$\sum V_{k-2}$	t_1	t_2
01_0200	4	4	139	2	10 194
05_0350	11	1	1683	1	5446
12_1500	2	57	1263	5384	5388
13_2000	3	165	2253	7345	7686
14_2500	5	168	4700	10 450	10 450
16_0260	11	358	3656	0	8543

Regarding the feasibility level, most of the k are proved optimal (25 of the 30 instances). Obviously, the instances for which no k^* is found are those belonging to the first subset described above. Now, we shall see what happens if we give more cpu time to the hybrid algorithm process.

Table 3 shows only the improved solutions reached after 3 computing hours. Unfortunately, no improvement has been obtained for the majority of the instances. Nevertheless, we can see again that it only concerns problems belonging to the first subset of `fapp` instances. Elsewhere, two new optimal values of k are proved (for the 12_1500 and the 13_2000 problems).

4.3. TABU *versus* AC-TABU

This section compares the behaviors of TABU alone and the hybrid *AC-Tabu* algorithm. These algorithms were executed 8 times with seed values varying from 0 to 7, and one hour of computing time.

Table 4 focuses on the k-feasibility level and gives the best ($k_{\min}$) and the worst ($k_{\max}$) solutions found by the TABU and hybrid *AC-Tabu* processes.

TABLE 4. Best and worst results for the level of feasibility.

	AC-Tabu		TABU			*AC-Tabu*		TABU	
fapp	$k_{\min}$	$k_{\max}$	$k_{\min}$	$k_{\max}$	fapp	$k_{\min}$	$k_{\max}$	$k_{\min}$	$k_{\max}$
01_0200	4	4	4	4	16_0260	11	11	11	11
02_0250	2	2	2	3	17_0300	4	4	4	4
03_0300	7	7	7	7	18_0350	8	8	8	8
04_0300	1	1	1	11	19_0350	6	6	6	9
05_0350	11	11	11	11	20_0420	10	10	10	10
06_0500	5	5	5	6	21_0500	4	4	4	5
07_0600	9	10	9	10	22_1750	7	7	7	11
08_0700	5	6	5	6	23_1800	9	9	9	11
09_0800	3	4	3	8	24_2000	7	7	7	11
10_0900	6	7	6	11	25_2230	3	3	10	11
11_1000	8	9	8	11	26_2300	7	7	7	10
12_1500	4	7	6	8	27_2550	5	5	11	11
13_2000	5	6	6	6	28_2800	3	3	9	11
14_2500	6	6	7	11	29_2900	6	6	8	12
15_3000	5	6	7	11	30_3000	7	7	11	11

The gap between $k_{\min}$ and $k_{\max}$ is higher for TABU, than for *AC-Tabu*, notably on the second subset of the benchmark. In conclusion, the behavior of *AC-Tabu* is more stable than that of TABU alone.

Note that the results of `fapp12_1500`, `fapp13_2000` and `fapp14_2500` do not contradict those from Table 3: it simply means that *AC-Tabu* finds a better solution in 3 hours with 0 random seed than in 8 × 1 hour with different random seeds.

4.4. COMPARISON WITH OTHERS METHODS

We will now present the results obtained during the ROADEF-2001 challenge, which took place in FRANCORO III, in the city of Québec (Canada).

The three compared algorithms (apart from ours) were developed by Caseau (MH+PPC), the Bisaillon team (Tabu), and the Michelon junior team (LNS+PPC).

The (MH+PPC) algorithm combines some constraint propagation with metaheuristics. More precisely, the algorithm increases the level, and at each one it shaves by strong consistency, shuffles by a Large Neighborhood Search (LNS) [14], which gives a non-feasible solution, and finally tries to obtain a feasible solution by a Limited Discrepancy Search (LDS) [10], such as a Variable Neighborhood local search (VNS) [9].

The second approach (Tabu) [7] is a local search approach based on *Tabu Search* that includes some original features, including a specialized neighborhood, heuristics to determine critical variables and values, different diversification techniques, an auto-adaptative mechanism to set the tabu list and finally, a pre-processing operation based on consistency techniques inherited from constraint programming. It is used to treat three different problems:

- Constraint Satisfaction Problem to find a solution at level k;
- the problem which satisfies all the constraints at level $k+1$, and as much as possible at level k;
- the problem which satisfies all the constraints at level $k+1$, and minimizes the objective function.

The Michelon junior team solves the *FAPP* by using a meta-heuristic based on a Large Neighborhood Search [14] and Constraint Propagation. Also, they relax certain 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 solution at this level k, and finally it improves this solution with respect to the two last points of the objective function. So as to compare several methods, in terms of quality, we have just given the three values of the objective function: k (underlined when the method proves optimality), the unsatisfied constraint number at level $k-1$, and the sum of the unsatisfied constraints at the levels below $k-1$.

The *AC-Tabu* column gives the results obtained by our method during the challenge. Note that they have been improved since the competition. Indeed, we have polished the code, and experiments are made on different computers. More details about the challenge, benchmarks and all the results can be found on the web site `http://www.prism.uvsq.fr/~vdc/ROADEF/CHALLENGES/2001/`

The main test condition is a computing time limited to one hour on a Pentium III, 500 Mhz, 128 Mo.

Regarding only the level k, MH+PPC found the optimal k 26 times, Tabu 27 times, LNS+PPC 27 times, and *AC-Tabu* 28 times. It seems that there is a problem with the LNS+PPC method, because in 4 instances, it could not find a solution ($k = 12$) and in several others it found no solution better than 11.

Regarding the two other components of the objective function, Tabu without AC gave the best results 27 times, although it could not prove optimality. In fact, the time spent on the PPC techniques in the other methods, was spent to improve the full objective function.

This method comparison reveals that the most difficult instances of the benchmark were `12_1500`, `13_2000` and `14_2500`.

To conclude, Table 5 highlights the effectiveness of approximate methods combined with some arc-consistent processes on very large instances.

Table 5. Comparison with three other methods.

	MH+PPC			Tabu			LNS+PPC			*AC-Tabu*		
fapp	k	V_{k-1}	$\sum V_{k-2}$	k	V_{k-1}	$\sum V_{k-2}$	k	V_{k-1}	$\sum V_{k-2}$	k	V_{k-1}	$\sum V_{k-2}$
01_0200	4	6	279	4	4	56	4	5	210	4	14	233
02_0250	2	18	248	2	7	86	11	1	435	2	20	195
03_0300	7	27	1076	7	10	341	11	1	1211	7	32	892
04_0300	1	164	0	1	31	0	2	1	282	1	184	0
05_0350	11	892	12 364	11	1	372	11	97	3459	11	364	5694
06_0500	5	53	1029	5	12	246	6	1	1086	5	31	811
07_0600	9	132	4419	9	22	714	12	-	-	9	106	3375
08_0700	5	53	1359	5	16	266	11	3	2144	5	73	1225
09_0800	3	63	937	3	28	195	4	2	999	3	104	846
10_0900	6	82	2365	6	18	475	11	4	3661	6	103	2003
11_1000	8	119	5206	8	8	1015	11	8	6146	8	119	4191
12_1500	7	180	6538	3	83	1698	11	647	13 797	2	62	1310
13_2000	7	229	7503	3	49	2003	11	671	15 145	5	132	3645
14_2500	8	18	10 661	4	35	3485	11	1209	24 751	5	217	5045
15_3000	7	333	9988	5	15	1569	11	1060	22 898	5	192	4727
16_0260	11	572	5779	11	5	56	11	590	5968	11	514	5189
17_0300	4	4	36	4	4	34	4	4	36	4	4	36
18_0350	8	4	55	8	4	55	8	4	57	8	4	59
19_0350	6	3	79	6	2	51	6	2	60	6	3	70
20_0420	10	6	145	10	5	97	10	5	106	10	7	142
21_0500	4	2	12	4	2	10	4	2	12	4	2	12
22_1750	7	16	356	7	15	187	7	15	292	7	25	503
23_1800	9	17	197	9	16	187	12	-	-	9	17	197
24_2000	7	7	90	7	6	71	7	6	77	7	9	91
25_2230	3	7	33	3	7	32	3	7	34	3	7	33
26_2300	7	10	81	7	9	74	7	9	75	7	10	86
27_2550	5	7	46	11	4	64	5	4	22	5	11	54
28_2800	3	32	129	3	13	32	3	14	72	3	42	142
29_2900	6	28	351	6	25	239	12	-	-	6	25	310
30_3000	7	17	602	11	1166	12 029	12	-	-	7	48	1045

5. Conclusion

In this paper we have presented a hybrid approach to solve the Frequency Assignment Problem with Polarization. *AC-Tabu* combines some arc-consistent techniques and an original *Tabu Search* process, with a large neighborhood exploration by means of maintaining consistency. It produces very good results on large-sized *FAPP* instances.

Tabu alone proves its own worth. Indeed, it provides most of the best results but at the same time, some of the worst ones. There are two main reasons that justify the filtering step: the behavior regularity of the hybrid search algorithm, and the proven optimality of the k-feasibility level, for most of the *FAPP* instances.

Some improvements to this approach are still to be expected. Two perspectives for future work can be identified. Firstly, making AC faster by studying the most recent AC algorithms [3] and adapting to the *FAPP* constraint semantic. Secondly, considering the two other objective function components V_{k-1} and $\sum_{i<k-1} V_i$ in the Tabu procedure.

Acknowledgement. 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 Gérard Verfaillie for fruitful discussion about some arc-consistent techniques.

References

[1] K.I. Aardal, C.A.J. Hurkens, J.K. Lenstra and S.R. Tiourine, Algorithms for Frequency Assignment Problems. *CWI Quartely* **9** (1996) 1-9.

[2] C. Bessière, Arc–consistency and Arc-consistency again. *Artif. Intell.* **65** (1994) 179-190.

[3] C. Bessière and J.C. Régin, Refining the Basic Consistency Propagation Algorithm, in *the 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, Seattle-Washington, USA (2001) 309-315.

[4] R. Dorne, *Étude des méthodes heuristiques pour la coloration, la T-coloration et l'affectation de fréquences*. Ph.D. thesis, Université Montpellier II Sciences et Techniques du Languedoc (mai 1998).

[5] C. Fleurent and J.A. Ferland, Genetic and Hybrid Algorithms for Graph Coloring. *Ann. Oper. Res.* **63** (1996) 437-461.

[6] P. Galinier, *Étude des métaheuristiques pour la résolution du problème de satisfaction de contraintes et de coloration de graphes*. Ph.D. thesis, Université Montpellier II Sciences et Techniques du Languedoc (janvier 1999).

[7] P. Galinier, S. Bisaillon, M. Gendreau and P. Soriano, Solving the Frequency Assignment Problem with Polarization by Local Search and Tabu, in *The 6th Triennal Conference of the International Federation of Operational Research Societies (IFORS'02)*, University of Edinburgh, UK (July 2002) 8-12.

[8] F. Glover and M. Laguna, *Tabu Search*. Kluwer Academic Publishers (1997).

[9] P. Hansen and N. Mladenovic, Variable Neighborhood Search. *Comput. Oper. Res.* **24** (1997) 1097-1100.

[10] W.D. Harvey and M.L. Ginsberg, Limited Discrepancy Search, in *Proc. of the 14th International Joint Conference on Artificial intelligence (IJCAI-95)* (1995) 607-613.

[11] M. Jiang, *Méthodes Approchées pour le Problème de Coloriage Généralisé Application au Problème d'Allocation de Fréquences Multiservices dans l'Aviation Civile*. Ph.D. thesis, Université Versailles Saint-Quentin-en-Yvelines (novembre 1996).

[12] A.K. Mackworth, Consistency in Networks of Relations. *Artif. Intell.* **8** (1977) 99-118.

[13] R. Mohr and T.C. Henderson, Arc and Path Consistency revisited. *Artif. Intell.* **28** (1986) 225-233.

[14] P Shaw, Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems, in *Principle and Pratice of Constraint Programming, CP'98* (October 1998).

[15] M. Vasquez, *Résolution en variables 0-1 de problèmes combinatoires de grande taille par la méthode tabou*. Ph.D. thesis, Université d'Angers, UFR de Sciences (December 2000).

[16] M. Vasquez and J.K. Hao, A "Logic-Constrained" Knapsack Formulation and a Tabu Algorithm for the Daily Photograph Scheduling of an Earth Observation Satellite. *Comput. Optim. Appl.* **20** (2001) 137-157.