CWI Quarterly Volume 9 (1 & 2) 1996, pp. 1 – 8

Algorithms for Frequency Assignment Problems

K.I. Aardal
Utrecht University, Department of Computer Science,
Padualaan 14, 3584 CH Utrecht, The Netherlands
e-mail: aardal@cs.ruu.nl

C.A.J. Hurkens, J.K. Lenstra, S.R. Tiourine
Eindhoven University of Technology, Department of Mathematics,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: {wscor,jkl,sergeyt}@win.tue.nl

Frequency assignment problems occur when a network of radio links has to be established. Each link has to be assigned an operating frequency from a set of available frequencies. The assignment has to satisfy certain interference limiting restrictions defined on pairs of links. Moreover, the number of frequencies used is to be minimized.

These problems have been investigated by a consortium consisting of research groups from Delft, Eindhoven, London, Maastricht, Norwich, and Toulouse. The participants developed optimization algorithms based on branch-and-cut and constraint satisfaction, as well as approximation techniques such as simulated annealing, taboo search, variable-depth search, genetic algorithms, and potential reduction methods. These algorithms were tested and compared on a set of real-life instances.

1. Frequency assignment problems

Given is a set L of radio links. Each link i is to be assigned a frequency f_i from a given domain D_i. Some links i may have a preassigned frequency p_i. Other restrictions are defined on pairs of links $\{i, j\}$: the frequencies of a pair of *interfering* links must be more than a given distance d_{ij} apart; the frequencies of a pair of *parallel* links must differ by exactly d_{ij}. Each link belongs to exactly one parallel pair. Some restrictions are *soft* and may be violated at a certain cost; the others are *hard* and may not be violated. Restrictions on the parallel links pairs are always hard. *Interference costs* c_{ij} for violating soft interference constraints and *mobility costs* m_i for changing soft preassigned

frequencies are given. An assignment of frequencies is *complete* if every link in L has a frequency assigned to it. We denote by C and M the sets of all soft interference and mobility constraints, respectively.

The *first order problem* is to find a complete assignment that satisfies all *hard* constraints at minimum cost:

$$\text{minimize} \sum_{C} c_{ij}\delta(|f_i - f_j| \leq d_{ij}) + \sum_{M} m_i\delta(|f_i - p_i| > 0)$$

subject to the hard constraints:

$$\begin{array}{rcll} |f_i - f_j| & > & d_{ij} & \text{for each pair of links } \{i,j\} \text{ with a hard interference constraint,} \\ |f_i - f_j| & = & d_{ij} & \text{for each pair of parallel links } \{i,j\}, \\ f_i & = & p_i & \text{for each link } i \text{ with a hard preassigned frequency,} \\ f_i & \in & D_i & \text{for each link } i, \end{array}$$

where $\delta(.)$ is 1 if the condition within brackets is true and 0 otherwise.

If there exists a *feasible* assignment, i.e., a complete assignment of zero cost, then the *second order problem* is to find a feasible assignment that satisfies *all* constraints and minimizes the number of distinct frequencies used:

$$\text{minimize } |\cup_i \{f_i\}|$$

subject to the hard and soft constraints:

$$\begin{array}{rcll} |f_i - f_j| & > & d_{ij} & \text{for each pair of interfering links } \{i,j\}, \\ |f_i - f_j| & = & d_{ij} & \text{for each pair of parallel links } \{i,j\}, \\ f_i & = & p_i & \text{for each link } i \text{ with a preassigned frequency,} \\ f_i & \in & D_i & \text{for each link } i. \end{array}$$

Complexity issues related to these problems are treated by HALE [7].

2. APPROXIMATION AND UPPER BOUNDS

Approximation algorithms seek to obtain good feasible solutions in a reasonable amount of time. They provide upper bounds on the minimum solution value.

Local search. Many approximation techniques use the principle of local search. The general idea is to start with an initial solution and iteratively perform small transformations of this solution in an attempt to improve the objective value. The *neighborhood* of a solution is defined as the set of solutions to which it can be transformed in one iteration. A mapping that specifies a neighborhood for each solution is called a *neighborhood function*.

A *search strategy* defines how to select a solution from a neighborhood. The basic *iterative improvement* strategy modifies at each step the current solution to a solution from its neighborhood of lower cost. It stops when no better neighbor exists. Other widely used search strategies include simulated annealing, taboo search and variable-depth search.

Straightforward implementations of these techniques [3, 4] that incorporate little structural information give reasonable solutions in moderate running times. More sophisticated implementations [8] that employ problem-dependent information in their neighborhood function and search strategy produce better results, often in less time.

Various hybrid approximation algorithms have been proposed, which combine local search with constructive, enumerative or iterative techniques. There exist constructive rules that apply local search to partial solutions, combinations of local search with partial enumeration or backtracking, and nested forms of local search. In the latter case, local search is applied at several levels; e.g., a neighbor obtained at one level is subjected to local search at a second level before the search at the first level is resumed.

Promising results were obtained using a network representation of the problem [3]. Each link corresponds to a cluster of nodes, with one node for each frequency in its domain. For each pair of interfering frequencies, there is an edge between the nodes in question. The mobility and interference costs define weights of the nodes and edges, respectively. A complete frequency assignment now corresponds to a subset of nodes, one from each cluster; its cost is the sum of the weights over the selected nodes and the edges induced by them. An assignment is subjected to iterative improvement, selecting a different node in some cluster at each step. When a local optimum is reached, it is made less attractive by increasing the weights of its nodes and edges, and the procedure is repeated.

Genetic algorithms. Genetic algorithms also apply local search but now with *hyperneighborhoods*, where a *set* of solutions is transformed into a new set of solutions. These methods usually mimic mechanisms from evolution theory. Starting from a *population* of random solutions, at each iteration its *offspring* is determined by applying *genetic operators*. The unary *mutation* operator and binary *crossover* operator make small random changes in the solutions in the population so as to obtain an offspring of better quality.

The trend that we observed for local search becomes even more marked for genetic algorithms. The approach must be tuned to the problem at hand in order to make it work, and then it can work very well [5, 9].

Incomplete optimization. Optimization algorithms usually apply some form of tree search. By limiting the search on heuristic grounds, one may gain speed, but at the expense of losing the optimality guarantee of the solution obtained. An attempt to use this approach in a *partial constraint satisfaction* algorithm produced poor results [2].

Potential reduction. An interior-point algorithm for binary feasibility problems was proposed by Karmarkar et al. in 1990. At first, one formulates the problem as an optimization problem, by dropping the integrality conditions and introducing a non-convex potential function, whose minimizers are feasible solutions to the original problem. Subsequently, an interior point method is used to obtain approximate solutions to the new problem. This algorithm starts with a

point in the interior of the feasible region and generates a sequence of interior points with decreasing objective value. For that purpose, at each iteration a descent direction for the potential function is determined using its quadratic approximation defined on an inscribed ellipsoid in the feasible region around the last generated point. The algorithm proceeds as long as a substantial reduction in the value of the potential function is found in that direction and no feasible integer solution has been generated by a rounding scheme. In case a local minimum is encountered, the potential function is modified in some way and the whole process is restarted.

A creative adaptation of this technique to frequency assignment problems [10] uses a specially structured binary quadratic formulation. Let x_{if} be equal to one if link i is assigned frequency f, and zero otherwise. The first order problem is now to

$$\begin{aligned} \text{minimize} \quad & x^T Q x \\ \text{subject to} \quad & \sum_{f \in D_i} x_{if} = 1, \quad i \in L, x \in \{0,1\}^{|L| \times |\cup_{i \in L} D_i|}, \end{aligned}$$

where Q is a matrix representing the interference and mobility costs. For the second order problem this formulation is slightly modified so as to incorporate an upper bound on the number of used frequencies. A potential function for these formulations has a sparse Hessian, which allows the use of sparse matrix techniques to facilitate an efficient implementation of the interior point algorithm. The method obtained fairly good results but has substantial memory requirements.

3. Optimization and lower bounds

Lower bounds on the minimum solution value are useful both in assessing the quality of approximate solutions and in limiting the search for the optimum. A lower bound is usually obtained by relaxing the problem, that is, by discarding some of its constraints and solving the remaining, simpler, problem.

Polyhedral lower bounds and branch-and-cut. If we wish to solve the frequency assignment problem to optimality, we need to use an enumerative algorithm such as branch-and-bound. To be able to solve real-life instances of such a computationally hard problem in this way, it is crucial to provide the algorithm with a very good lower bound. The lower bound provided by the linear relaxation of the problem, which is obtained by dropping the integrality requirements on the variables from the original formulation, is too weak. *Polyhedral lower bounds* are obtained by identifying additional linear inequalities that are necessary in the linear description of the convex hull of feasible solutions. Such inequalities are called *facet defining*. If we would know the linear description completely, then we could solve the problem as a linear programming problem, which is computationally easy. Obtaining such a complete description is, however, as hard as solving the problem itself. We therefore settle for certain families of linear inequalities that define facets of the part of the convex hull

of solutions where we expect to find the optimal solution. Since the formulation should not grow too big, we add inequalities only if they are violated by the current fractional solution. An algorithm that at each iteration adds violated linear inequalities to the current formulation, and resolves the linear optimization problem based on the new extended formulation is called a *cutting plane algorithm*. In *branch-and-cut* we apply a cutting plane algorithm in every branch-and-bound node. If we just need a lower bound to measure the quality of a feasible solution, we can use the bound obtained in the root node of the search tree.

The classes of facet-defining inequalities used for the first order problem are all variants of the general class of *clique inequalities*. These inequalities are based on cliques (complete subgraphs) in the graph $G = (V, E)$ obtained by introducing one vertex for every link, and an edge between vertices i and j if $d_{ij} > 0$. The basic clique inequality is of the following form. Let C be a clique in G. Since $d_{ij} > 0$ for all pairs of vertices $i, j \in C$, at most one link i corresponding to a vertex in C can be assigned a specific frequency f. The inequality $\sum_{i \in C} x_{if} \leq 1$ is therefore valid. Since such an inequality is facet defining only if the clique C is maximal, we identify maximal cliques in G. In general this problem is hard as well, but since the maximal cliques in the graphs corresponding to our instances are of modest size, this can be done efficiently. To improve the performance of the cutting plane algorithm it is important to use logical implications to increase the size of the cliques in G. This is one of the main ingredients of *preprocessing*, which is carried out before the branch-and-cut algorithm is called.

An algorithm of this type solved all feasible test instances to optimality in a reasonable amount of time [1].

Constraint satisfaction. Constraint satisfaction is another specialized branch-and-bound algorithm. It is a technique for solving integer feasibility problems. Given a set of constraints, it tries to find a solution that satisfies them, or to conclude that no such solution exists. At each node of the search tree a variable is chosen and assigned a value from its domain. Special selection rules are employed to choose an appropriate variable and its value in order to reduce the depth of the tree. Consistency enforcing techniques are applied to eliminate values from the domains of the free variables that are inconsistent with the partial assignment made. If at some node of the tree the domain of a free variable becomes empty, this node is fathomed and backtracking is performed to the nearest active node.

A problem-specific implementation of constraint satisfaction for the second order problem [9] succeeded in finding provably optimal solutions to all test instances.

Ad hoc lower bounds. To obtain a lower bound for the first order problem, a relaxation is considered that, instead of assigning a frequency to each link, decides on whether or not to adhere to its preassigned frequency. For this relaxation it is easy to define an objective function that covers most of the incurred

costs and is subject to integrality constraints only. Denote the interference cost of a pair of links $\{i, j\}$ by K_{ij} if i and j are set to their preassigned frequencies:

$$K_{ij} = \begin{cases} c_{ij}, & \text{if } |p_i - p_j| \leq d_{ij} \ , \\ 0, & \text{otherwise,} \end{cases}$$

and by K_{ifj} if frequency f is assigned to i, while j is set to its preassigned frequency:

$$K_{ifj} = \begin{cases} c_{ij}, & \text{if } |f - p_j| \leq d_{ij} \ , \\ 0, & \text{otherwise.} \end{cases}$$

Let the decision variables x_i be equal to one if link i is set to its preassigned frequency, and zero otherwise. The relaxation can be formulated as:

$$\begin{aligned} \text{minimize} \quad & \sum_{i,j \in L, i<j} K_{ij} x_i x_j + \\ & \sum_{i \in L} m_i (1 - x_i) + \sum_{i \in L} (1 - x_i) (\min_{f \in D_i \setminus p_i} \sum_{j \in L} K_{ifj} x_j) \\ \text{subject to} \quad & x_i \in \{0, 1\} \ \ \forall i \in L. \end{aligned}$$

A preprocessing technique and an efficient enumeration scheme incorporated in a branch-and-bound framework were used to obtain tight lower bounds for test instances with nonzero mobility costs [8].

4. Comparison

Table 1 presents the computational results obtained on a set of real-life instances that vary in size between 200 and 916 links. For the feasible ones, which allow an assignment that satisfies all soft and hard constraints, a second order problem has to be solved. For the infeasible instances, the cost of violating the soft constraints is to be minimized. The values of interference and mobility costs in these instances vary in a wide range. For each approach we indicate the quality of the reported solutions and an average running time for the classes of feasible and infeasible instances. The list of computer equipment used should help to correct running times for differences in hardware performance.

5. Conclusions

Frequency assignment problems form a new class of relevant problems, to which many of the standard techniques of combinatorial optimization can be applied. Each of these techniques has its benefits. Overall, there is a strong positive correlation between the amount of problem-specific information used, the extent to which mathematical insight is exploited, the development and implementation effort required, and the quality of the results obtained.

method	group	feasible instances							infeasible instances					
		1	2	3	4	5	11	time	6	7	8	9	10	time
SA (simulated annealing)	EUT	2	0	0	0	0	2	1	7%	65%	5%	0%	0%	310
TS (taboo search)	EUT	2	0	0	0	–	0	5	–	–	–	–	–	–
VDS (variable-depth search)	EUT	2	0	0	0	–	2	6	4%	0%	14%	0%	0%	85
SA (simulated annealing)	CERT	4	0	0	0	–	10	41	43%	1299%	70%	2%	0%	42
TS (taboo search)	KCL	2	0	0	0	0	2	40	168%	1804%	566%	8%	1%	111
GENET (see Section 2)	KCL	0	0	0	0	0	2	2	13%	27%	40%	–	–	20
GA (genetic algorithm)	UEA	6	0	2	0	i	10	24	1%	386%	134%	3%	0%	120
GA (genetic algorithm)	LU	–	–	–	–	–	–	–	0%	0%	0%	0%	0%	hrs
PCS (partial constraint satisfaction)	CERT	4	0	6	0	0	–	28	84%	2563%	246%	47%	12%	6
PR (potential reduction)	DUT	0	0	2	0	0	–	3+	28%	–	–	4%	1%	10+
BC (branch-and-cut)	DUT,EUT	0*	0*	0*	0*	0*	0*	<10+	–	–	–	–	–	–
CS (constraint satisfaction)	LU	0*	0*	0*	0*	0*	0*	hrs	–	–	–	–	–	–
LB (ad hoc lower bounds)	EUT	−2	0	−2	0	–	−2	min	–	–	–	−5%	−1%	hrs
best solution		16*	14*	14*	46*	792*	22*		3389	343592	262	15571	31516	

feasible instances
2 number of frequencies in addition to the optimal solution
i infeasible solution is reported

infeasible instances
7% deviation from the best known solution

– the method is not applicable, or no solution is reported
∗ optimality of the solution is proved

computer hardware
CERT SUN SPARC 10
DUT HP9000/720
EUT SUN SPARC 4
KCL DEC Alpha 3000 (130 MHz)
LU (GA) DEC OSF/1 AXP
LU (CS) PC
UEA DEC Alpha (133MHz)

running times
310 minutes of computation time
min minutes of computation time
hrs hours of computation time
+ preprocessing time is not included

TABLE 1. Computational results

We refer to the full paper for a more detailed presentation and analysis of the algorithms and their results.

ACKNOWLEDGEMENTS

Our research was supported by the EUCLID program, CEPA 6 (Artificial Intelligence), RTP 6.4 (Combinatorial Algorithms for Military Applications) [6]. We are grateful to the Centre d'Electronique de l'Armement and Delft University of Technology for providing test instances, and to all participants in the CALMA project for providing detailed information about their methods and results.

REFERENCES

1. K.I. AARDAL, A. HIPOLITO, C.P.M. VAN HOESEL, B. JANSEN (1994). *A branch-and-cut algorithm for the frequency assignment problem*, Manuscript.
2. E. BENSANA, T. SCHIEX. *Partial constraint satisfaction*, CALMA Technical Report 2.2.3, Centre d'Etudes et de Recherches de Toulouse, France.
3. A. BOUJU, J.F. BOYCE, C.H.D. DIMITROPOULOS, G. VOM SCHEIDT, J.G. TAYLOR. *Tabu search and GENET*, CALMA Technical Report 2.3.4, Centre d'Etudes et de Recherches de Toulouse, France (also available at `http://www.win.tue.nl/~wscor/calma.html`).
4. P. BOURRET. *Simulated annealing*, CALMA Technical Report 2.3.1, Centre d'Etudes et de Recherches de Toulouse, France.
5. P. CHARDAIRE, A. KAPSALIS, J.W. MANN, V.J. RAYWARD-SMITH, G.D. SMITH (1995). Applications of genetic algorithms in telecommunications. *Proc. Applications of Neural Networks to Telecommunications* **2**, Stockholm, May 1995. J. ALSPECTOR, R. GOODMAN, T.X. BROWN (eds.), Lawrence Erlbaum, 290-299.
6. W. HAJEMA, M. MINOUX, C. WEST (1993). *Statement of the radio link frequency assignment problem*, Request for proposals on the CALMA project, Technical appendix.
7. W.K. HALE (1980). Frequency assignment: theory and applications, *Proc. IEEE 68*, 1497-1514.
8. C.A.J. HURKENS, S.R. TIOURINE (1995). *Upper and lower bounding techniques for frequency assignment problems*, Memorandum COSOR 95-34, Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, The Netherlands.
9. A.W.J. KOLEN. *Constraint satisfaction*, CALMA Technical Report 2.2.2, Centre d'Etudes et de Recherches de Toulouse, France (also available at `http://www.win.tue.nl/~wscor/calma.html`).
10. J.P. WARNERS (1995). *A potential reduction approach to the radio link frequency assignment problem*, Master's Thesis, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, The Netherlands.