Feasible Cellular Frequency Assignment Using Constraint Programming Abstractions

Joachim P. Walser
Programming Systems Lab
Universität des Saarlandes
Geb. 45, Postfach 151150
66041 Saarbrücken, Germany
walser@ps.uni-sb.de

Abstract

The contribution of this paper is twofold. We present a new method for feasible cellular frequency assignment, a hard combinatorial optimization problem from telecommunications. Frequency assignment problems arise when a cellular radio network has to be established. Given a number of base stations, the goal is to assign each a number of frequencies, subject to given interference restrictions. We develop a transformation technique that allows for approximate optimization of multiple criteria: First, the original problem is transformed, thereby reducing the allowed number of distinct frequencies. In a second stage, the frequency span is compressed. Both stages exploit the cell-structure of the problem formulation. Preliminary experiments on randomized problems examine the effectiveness of the approach with respect to both criteria.

As we proceed in solving the subproblems that arise, we identify certain key programming abstractions (such as constraints, propagation and search). We argue that if these abstractions are supported by a programming language, they can greatly speed up the search for an efficient algorithm. We exemplify certain aspects of the modelling in Oz, a higher-order concurrent constraint language.

Keywords: cellular frequency assignment, constraints, search, Oz

1 Introduction

Frequency assignment problems arise when a cellular radio network has to be established. There are many types of frequency assignment problems that may involve positioning radio transmitters or allocating frequencies for an existing network of transmitters (see [8] for an overview). We consider a problem that has been classified as 'frequency constrained channel assignment problem' [8] in a formulation that is structured according to transmitter cells [12, 3, 10]. In each cell, a base station broadcasts a number of channels. Channels may interfere, thus their assigned frequencies may require a minimal distance from one another. The problem consists of assigning a frequency to each channel, subject to the interference constraints.

We will refer to this sort of frequency assignment problem as *cell-oriented*: Each cell contains *several* channels and all interference constraints are stated on cells and apply to all the channels within a cell. We present a two-stage approach with the objectives to first *limit* the number of different frequencies in possible solutions (through frequency reuse) and second to *assign* frequencies such that the resulting span will be small. Both stages operate on the cell structure rather than on the channel structure. Like [12], we show that if the problem can be stated in terms of cells with each cell requiring several frequencies, this cell structure can be exploited.

The paper is structured as follows. Section 2 gives a formalization of the problems being solved. Section 3 presents *iterative clique covering*, a preprocessing stage which transforms the problem, essentially by posting equality constraints for non-interfering channels. These additional constraints limit the number of different frequencies in the solutions. By operating on the cell structure, the complexity is reduced from coloring a *channel* graph to coloring a *cell* graph, which can significantly reduce the problem size. Section 4 presents the second stage, *cell sequencing*, which assigns frequencies to the preprocessed problem with the objective to minimize the frequency span. The method is (theoretically) complete and avoids symmetries between channels within one cell. Together, the two stages can approximately optimize both criteria. Figure 1 illustrates the scenario.

Figure 1: Two-stage approximate optimization for the frequency assignment problem (FAP).

For each stage, we identify key programming abstractions that were used for its implementation in the concurrent constraint language Oz [17, 16]. We argue that by providing abstractions like constraints, propagation and search, constraint languages in general and Oz in particular encourage the prototyping of new ideas.

2 Problem Formulation

We consider a type of frequency assignment problem that is stated in terms of transmission *cells* [12, 3, 10, 8]. There are k cells in $V \subset \mathbb{N}$. Each *cell* i contains n_i *channels* (we also say cell i has a *demand* of n_i frequencies). The total number of channels is $M = \sum_i n_i$. Each channel l from cell i is to be assigned a frequency f_{il} from a global domain D_f, subject to given *interference limiting constraints* defined on pairs of cells: The frequencies of two channels l and m of cells i and j must be at least a given distance d_{ij} apart, $|f_{il} - f_{jm}| \geq d_{ij}$, $l = 1 \dots n_i$, $m = 1 \dots n_j$; we assume a symmetric $k \times k$ interference matrix $D = (d_{ij})$. If there is no interference between i and j then $d_{ij} = 0$. Typically, co-site interference is $d_{ii} > 0$. A triplet $F = (V, N, D)$ will be called a *frequency assignment problem* (FAP) if V is a set of cells, $N = (n_i)$ is a demand vector and $D = (d_{ij})$ is an interference matrix. A *solution* of F is an assignment of frequencies from the domain D_f to all channels $f_{il}, i = 1 \dots k, l = 1 \dots n_k$ violating no interference constraints.

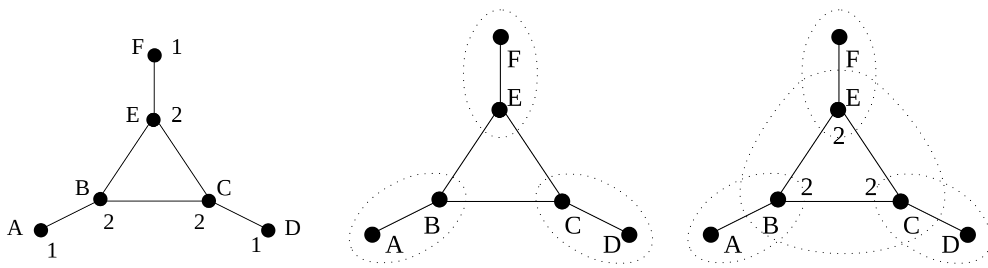
We will refer to the *interference cell graph* $G = (V_G, E_G)$, as the graph whose vertices $V_G = V$ are the transmission cells, and whose edges E_G are connecting cells. Cells $\{i, j\}$ are connected by an edge if they *interfere* ($d_{ij} > 0$). The *inverted cell graph* is $\bar{G} = (V_G, V_G \times V_G \setminus E_G)$.

Sometimes, not all constraints are *hard*, but may be violated at a certain interference cost [1, 3]. The *first order problem* is to find a complete assignment of frequencies to channels that satisfies all *hard* constrains at minimum overall cost. 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 minimizes one of the following criteria [8]:

1. the *frequency order* (the number of distinct frequencies):
   minimize $|\bigcup\{f_{il} \mid i = 1 \dots k, l = 1 \dots n_i\}|$

2. the *frequency span*: minimize $\max\{f_{il} \mid i = 1 \dots k, l = 1 \dots n_i\}$
   (without loss of generality we assume that frequency 1 is always assigned.)

We will only consider the second order problem here and present an approximation algorithm for what has been called “the minimum span of a minimum-order feasible assignment” [8]. We



(a) An example of an inverted cell graph $\bar{G}$ with cells A through F, having frequency demands 1 or 2.

(b) An optimal clique covering for $\bar{G}$, which covers each cell only once.

(c) An optimal clique covering for $\bar{G}$. If a cell i demands n_i channels, it is covered n_i times.

Figure 2: Iterative clique-covering (ICC) as approximate minimization technique for the number of distinct frequencies.

note that in some realistic formulations of the FAP [1], channels may have different domains, or may have preassigned frequencies. Some of these additional constraints may be introduced at a later stage of this work.

3 Reducing the Frequency Order

The traditional approach to minimizing the frequency order is to perform graph coloring on the *channel* graph [19, 8, 3, 11]. Because each color can be identified with one frequency, the minimal number of colors (the chromatic number of the graph) equals the minimal number of frequencies needed. Coloring this channel graph, however, is non-trivial since there may be more than 150 vertices and the chromatic number may exceed 100.

Penotti and Boorstyn [12] have first recognized the close connection between the channel graph and the cell graph and apply graph coloring of the cell graph to *reduce* the input problem. Our approach similarly exploits the cell structure, however we use it for a *transformation* of the problem. The approach iteratively identifies a group of non-interfering cells, picks one channel from each cell of the group and posts equality for all those channels. This grouping of channels is used to transform the original problem to a problem in which the number of different frequencies is bounded above. The question is *which* cells should iteratively be grouped together so as to result in a minimal number of distinct frequencies? We do not know of an optimal deterministic grouping strategy but will present an approximation strategy that appears to do well in our examples.

To find such a group of cells, the vertices of the interference cell graph can be colored, subject to the constraints that two *adjacent* vertices be colored differently (since the cells interfere). This equivalently amounts to partitioning the inverted cell graph $\bar{G}$ into a minimal number of *cliques*[1]. The covering strategy shall be explained by means of an example. Figure 2 (a) shows an inverted cell graph $\bar{G}$ of cells A through F. In $\bar{G}$ two vertices (cells) are connected by an edge if and only if there is *no* interference between them. The numbers refer to the frequency demand for each cell. We assume that channels within the same cell interfere, i. e. $d_{ii} > 0$ (otherwise, they would collapse). We start by partitioning $\bar{G}$ into a minimal number of cliques, the result of which is shown in (b). Of this partition, a random largest clique is considered, say {A,B}. One channel from each cell A and B is removed (thereby reducing the demands of A and B by one) and

[1] A clique in a graph is a completely connected subgraph.

the two channels are constrained to be equal. Then, the remaining graph is colored afresh, i. e. all the cells in $\bar{G}$ still containing channels are partitioned into cliques. This time, there are two minimal partitions: one that includes the clique $\{B, C, E\}$, and one that doesn't. Thus, there are different possible paths to complete the covering. Those steps of clique partitioning and posting equality constraints for the largest clique are iterated until no more cells are left in the graph. In this example, no matter which largest clique is picked in each step, the result will be (c). Each clique corresponds to one frequency, and (c) happens to be optimal, i. e. all remaining solutions have only four colors. In general, if some cells demand more than one channel, one can construct examples for which ICC will not lead to an optimal solution, however it is not clear how often such instances appear as practical problems. Because the above strategy will iteratively cover the cell graph with cliques, we refer to it as *iterative clique covering* (ICC).

Thus far, ICC was described as preprocessing stage in a constraint satisfaction approach. Since the span-compression technique in the subsequent stage prefers a plain cell structure as input (with no equality constraints added), we will take a slightly different approach now. We describe ICC as a transformation on the cell graph that introduces new cells that contain combinations of original channels. The solution of the original problem can be obtained by a simple inverse-transformation.

Algorithm 1 *Iterative clique covering* (ICC)

Input: *A frequency assignment problem $F = (V, N, D)$, given by a non-empty set of cells $V \subset \mathbb{N}$, a frequency demand vector $N = (n_i)$ and a symmetrical interference restriction matrix $D = (d_{ij})$ for pairs of cells.*

Output: *A transformed frequency assignment problem $\hat{F} = (\hat{V}, \hat{N}, \hat{D}), V \subseteq \hat{V}, D$ a sub-matrix of $\hat{D}$, and revised frequency demands $\hat{N}$.*

1. *Initialize $\hat{F}$ with $\hat{V}, \hat{D}, \hat{N}$ as V, D and N. Let $G = (V_G, E_G)$ be the interference cell graph induced by F. $V_G := V, E_G = (e_{ij})$, $e_{ij} := 1$ iff $d_{ij} > 0$.*

2. *Find any minimal coloring of G, such that any two adjacent vertices are colored differently. This coloring directly induces a partition of the inverted graph $\bar{G}$ into cliques: $P = \{C_1, \ldots, C_n\}$.*

3. *Pick any maximal clique C_m from P (i. e. for any $C_i \in P : |C_i| \leq |C_m|$). Combine $\hat{n}_p := \min_{v \in C_m} \hat{n}_v$ channels from each cell of C_m in a fresh cell p in $\hat{V}$. Adjust the new demands of all cells whose channels have moved: $\hat{n}_i := \hat{n}_i - \hat{n}_p$, $i \in C_m$. Remove every cell i from G whose frequency demand $\hat{n}_i$ has become zero.*

4. *For the new cell p, define interference restrictions according to the maximal restrictions of all cells in C_m: Extend $\hat{D}$, setting interference between p and any $i \neq p$ to $\hat{d}_{pi} := \hat{d}_{ip} := \max_{v \in C_m} \hat{d}_{vi}$. The co-site interference is $d_{pp} := \max_{v \in C_m} d_{vv}$.*

5. *If there are cells left in G, goto 2, else return $\hat{F}$.*

Remarks: Note that in the algorithm several channels are combined within a new cell in *one* step. $\hat{V}$ will contain the new cells to hold channel combinations, plus all cells of V (some with 0-demands in $\hat{N}$, if all channels have been moved to new cells).

Programming Abstractions For programming languages not supporting constraints, the ICC algorithm may get fairly involved since it embeds graph coloring, for which elaborate techniques are needed. On the other hand, its implementation in a constraint programming language like CHIP [4] or Oz [17, 16] is straightforward because the necessary abstractions are provided. For the graph-coloring task (2), our implementation uses the following abstractions: Inequality constraints are posted for interfering cells, and constraint propagation (arc consistency) is employed as implemented in Oz's Finite Domain module [13]. An off-the-shelf branch-and-bound search algorithm is used together with a labelling strategy described in [5]. Node ordering was implemented according to a first-fail heuristic. With these building blocks, efficient graph coloring for graphs of moderate size is possible. The graph coloring can easily be embedded in the iterative algorithm.

4 Compressing the Frequency Span

In this section we develop a method that is suited for finding frequency assignments with a minimum frequency span. If there are a total of M channels to be assigned frequencies from a domain $\{1, \ldots, D_f\}$, there are $(D_f)^M$ possible combinations (e.g. $50^{64} \approx 10^{108}$ for 64 channels from a spectrum of span $D_f = 50$). One observation is that for a minimization of the frequency span, many potential assignments will surely not be optimal because frequencies may still be compressed. Additionally, many solutions are symmetrical because all channels within one cell are equivalent. In the following, we will give a new approach to span minimization, which avoids symmetries and many non-optimal solutions.

4.1 The Cell-Sequencing Approach

To reduce the search space, we will propose a problem encoding that is inspired from task ordering in the job-shop scheduling domain [2]. In task ordering, the idea is the following: Rather than directly assigning start times to the given tasks, one orders the tasks that are competing for the same resource. Since a task ordering induces start times for the tasks, this is a constructive approach. The encoding has been shown to greatly reduce search space size in the job-shop scheduling domain [2, 15].

Similarly, one can view the frequency assignment problem as the problem of *sequencing* cells. By a *cell sequence*, we mean a sequence of cells $(c_1, c_2, \ldots), c_i \in V$. A cell sequence $(c_1, \ldots, c_M)$ in which each cell i appears exactly n_i times will be called *satisfying*.

A cell sequence induces a unique frequency assignment because for each occurrence of a cell, an unassigned channel can be picked and be assigned a *smallest* possible frequency, subject to the (hard) interference constraints (d_{ij}). If the sequence is satisfying, the frequency assignment will be complete. We now explain how an admissible frequency assignment[2] is induced by a cell sequence. Let $(c_1, c_2, \ldots, c_l)$ be the cell sequence thus far; we define the mapping F to frequencies inductively. Let $F(c_1) = f_1 = 1$. Now let $F(c_1, c_2, \ldots, c_l) = (f_1, f_2, \ldots, f_l)$ be the frequencies assigned to the cells up to this point. Let t_i be the largest frequency that has been assigned to cell i, $t_i = max_{1 \leq j \leq l}\{f_j \mid c_j = i\}$; if i is not in the sequence, let $t_i = 0$ (0 being no valid frequency). In the next step, cell c_{l+1} is to be assigned the smallest possible frequency obeying the (hard) interference constraints with all other cells:

$$f_{l+1} = \max_{1 \leq i \leq k}\{t_i + d_{i,l+1} \mid d_{i,l+1} > 0 \wedge t_i > 0\}$$

Any assignment of a greater frequency to cell c_{l+1} would surely result in a non-optimal span for the sequence $(c_1, \ldots, c_{l+1})$. Thus, many non-optimal assignments will be avoided by cell sequencing. Another property of cell sequencing is that symmetries between channels within the same cell are avoided, because any unassigned channel can be picked.

With cell sequencing, span optimization amounts to finding a satisfying cell sequence $(c_1, \ldots, c_M)$ for which the largest assigned frequency is minimal. Clearly, in each step one can only choose

[2] An *admissible* assignment is a mapping from f_{il} to $\mathbb{N}$, $i = 1 \ldots k, l = 1 \ldots n_i$.

from all cells that are not yet satisfied. It is possible to show that there always exists a satisfying sequence that results in an optimal span; the completeness prove will be given elsewhere.[3]

The Search Space With cell sequencing, a solution to the frequency assignment problem consists of a permutation $(c_1, c_2, \ldots, c_M), c_i \in V$. There are $M = \sum_{1 \leq j \leq k} n_j$ channels, thus there are $\binom{M}{n_1\ n_2\ \cdots\ n_k}$ cell permutations. The overall frequency domain does not affect the size of the search space. To give an example, for 8 cells, each requiring 8 frequencies, there are $\frac{64!}{(8!)^8} \approx 10^{52}$ permutations – though considerably less than D_f^M ($\approx 10^{108}$ with $D_f = 50$), there are still too many to be searched exhaustively.

Heuristics The remaining problem is to construct good satisfying cell sequences that will lead to a small span. A heuristic approach we found to work well for our examples is *small-frequencies-high-demand-first*: In each step, among the cells that have non-zero demands, choose a cell that will result in the smallest new frequency. Ties are broken by considering the remaining demand, and cells with higher demand are chosen first.[4]

Search Thus far, the approach is essentially greedy. However, we can extend it by introducing choice points. At each point, there is choice between different cells, so the heuristic estimation imposes an order of which choices to explore first. In addition to the local heuristic preference, there is freedom in the *global* search strategy. In depth-first search one follows the heuristic all the way down to a leaf, backtracking to the alternatives in the left corner of the tree. In contrast, one can explore the search tree in more sophisticated ways: Harvey and Ginsberg [9] present an intuitively appealing search strategy called limited discrepancy search (LDS): "The idea is that a small number of wrong turns can be overcome by systematically searching all paths that differ from the heuristic path in at most a small number of decision points, or 'discrepancies'. Limited discrepancy search is a backtracking algorithm that searches the nodes of the tree in increasing order of such discrepancies." LDS(0) follows the heuristic in all decisions. LDS(1) follows the heuristic except once in the path, and so on. In our experiments we apply LDS(0) followed by LDS(1), and report the number of probes down to a leaf.

Relation to Existing Approaches Cell sequencing is related to an early assignment strategy called *frequency-exhaustive* assignment [19, 7]. Cell sequencing contributes a new frequency-selection strategy ('holes' are not tried to fill), and with search, it is (theoretically) an optimal method. It avoids symmetries and applies a recent search strategy, LDS, which can compensate for a small number of wrong heuristic decisions.

Programming Abstractions It is fairly straightforward in any kind of language to implement a greedy approach of the cell-sequencing strategy with a small-frequencies-first heuristic. However, as it comes to search, in conventional Prolog-based constraint languages (e. g. clp(FD), CHIP, Eclipse) one is limited to the built-in depth-first search strategies, and implementing other strategies is only possible by by-passing chronological backtracking. In conventional approaches in imperative, logic or functional languages one will merge the search strategy with the solving algorithm, and thus not have a clean separation.

The concurrent constraint language Oz takes a different approach in which search is orthogonal to the problem formulation [14]: One can generate choice points in a functional or constraint program, and in a separate module a search strategy takes responsibility to execute them in any

[3] Note that with the ICC preprocessing, completeness of cell-sequencing for the original problem is no longer guaranteed.

[4] Other heuristics, e. g. the ones described in [7] have to be evaluated empirically at a later stage.

programmable order. Alternatively, one of several predefined strategies can be chosen. Since Oz supports programmable search [14], strategies like LDS can be coded in Oz almost identically to the original abstract formulation and subsequently be used off-the-shelf. Different strategies can easily be compared experimentally, allowing for fast prototyping.

5 Experimental Results

In this section we give preliminary experimental results for a sample of randomly generated frequency allocation problems. All instances are cell oriented, i.e. several channels are to be allocated within each cell. The appendix contains FAP2 and FAP5. At present, we are not able to compare our results with other approaches than a naive constraint propagation approach (described below) since we are not aware of other approaches that aim at minimizing both frequency order and span.

problem instance	M	ICC & sequencing (order,span)	lb order	lb span (untransf.)	LDS probes	naive (order,span)
FAP1	148	(95,145)	95	145 (2%)	13	(95,165)
FAP2	165	(74,156)	69	146 (7%)	11	(97,192)
FAP3	150	(86,137)	83	131 (5%)	12	(97,164)
FAP4	161	(91,159)	90	146 (9%)	37	(95,178)
FAP5	222	(122,274)	104	273 (0%)	2	(137,327)
FAP6	222	(222,222)	222	222 (0%)	1	

Table 1: Experimental results for cellular frequency assignment on randomly generated problems, each 25 cells. Column 2 reports the number of channels (M), column 3 gives the results. Column 4 reports lower bounds on the order, and column 5 gives span lower bounds and percent deviation from them (note that these bounds are computed for the untransformed problems). Column 6 reports the number of LDS(0)+LDS(1) probes for the reported span, the last column gives the naive results. All execution CPU times in Oz-2.0 were around 40s for 40 LDS-probes on a SPARCstation 20 (502,160MB) except FAP5 and FAP6 with 110s.

Problem Descriptions: Problem FAP1 originates from [10] and requires around 8000 distance constraints. All other problems were generated randomly. FAP1 and FAP2 are 70% and 80% and FAP3–5 are 85% constrained (percent of cell pairs interfering). FAP4 and 5 have an additional clique built in that helps to increase the order lower bound. Instances 2 through 5 are available on the Web at `http://ps.uni-sb.de/~walser/fap/fap.html`. Please note that without prior ICC transformation, cell sequencing was able to find solutions 0-1% within the span lower bounds. Since there exists a tradeoff between minimal order and minimal span, in general one cannot expect to achieve optimal results with respect to both criteria simultaneously.

The naive strategy posts all distance constraints and applies a simple labelling strategy with branch-and-bound search, the best solution in 10k nodes is reported. By chance, the naive propagation yields the solution of FAP1 without any search.

Lower Bounds: We computed the lower bound on the order following an idea reported in the Calma Euclid project [18]. A maximum clique is identified in the cell interference graph and one frequency is allocated for each channel within the cells in the clique. The lower bound on the span is essentially the first-level bound $\max_i (n_i - 1)\, d_{ii} + 1$ with additional consideration if several connected cells had the same highest frequency demand.

6 Conclusion and Future Work

We have studied a cell oriented variant of the feasible frequency assignment problem. Our aim in this paper has been to present a two-stage approach that approximately optimizes the two criteria frequency order and frequency span. We have shown a way how cell structure can be exploited, provided it is present in the encoding. By operating on cells rather than channels, the complexity can be reduced and symmetries and many non-optimal solutions can be avoided. For an implementation of the algorithms in a concurrent constraint language like Oz, we have identified certain key programming abstractions. The intent was to show that these abstractions, if supported by a programming language, can lead to a clearer factorization of the problem (into constraints, propagation and search) and thus lead to shorter development times.

In this paper we focussed on describing the assignment algorithms, so obviously much remains to be done. First of all, one needs to evaluate the presented techniques on realistic problems, and competing algorithms have to be identified for this problem class. Second, we would like to examine theoretical properties of ICC with the hope to identify a class of FAPs for which the transformation produces optimal order. Also, we would like to evaluate other heuristics previously reported in the literature [19, 7]. Additionally, one could model the problem in a pure constraint approach by posting equality constraints in the first stage and constructing appropriate labelling strategies in the second. The hope would be that this formulation could handle additional constraints like pre-assigned frequencies. Last, extending cell sequencing to handle soft constraints would be interesting. As cellular Radio suppliers are typically purchasing a band of frequencies [6] there is a cost introduced by the frequency span and a tradeoff exists between the overall interference (small span) and the cost (wide span). We expect that the cell sequencing approach extended with soft constraints could be used to vary this tradeoff.

7 Acknowledgements

This work was supported by a doctoral fellowship of the Deutsche Forschungsgemeinschaft (DFG) to the author (Graduiertenkolleg Kognitionswissenschaft). The author is grateful to Jörg Würtz and Mats Carlsson for helpful discussions and comments on earlier versions of this paper. Many thanks also to Martin Müller, Joachim Niehren and Christian Schulte for helpful discussions of this work.

References

[1] Karen Aardal, C.A.J Hurkens, J.K. Lenstra, and S.R. Tiourine. An overview of algorithmic approaches to frequency assignment problems, calma project. Technical report, Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, The Netherlands, 1995.

[2] J. Carlier and E. Pinson. An algorithm for solving the job-shop problem. *Management Science*, 35(2):164–176, 1989.

[3] Mats Carlsson and Mats Grindal. Automatic frequency assignment for cellular telephones using constraint satisfaction techniques. In *Proceedings of the Tenth International Conference on Logic Programming*, 1993.

[4] M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf, and F. Berthier. The constraint logic programming language CHIP. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 693–702, Tokyo, Japan, 1988.

[5] Mehmet Dincbas, Helmut Simonis, and Pascal Van Hentenryck. Solving large combinatorial problems in logic programming. *Journal of Logic Programming*, 8:75–93, 1990.

[6] Andreas Eisenblätter, 1996. Personal communication.

[7] Andreas Gamst and Werner Rave. On Frequency Assignment in Mobile Automatic Telephone Systems. In *Proc. GLOBECOM*, pages 309–315. IEEE, 1982.

[8] William K. Hale. Frequency assignment: Theory and applications. In *Proceedings of the IEEE*, volume 68, pages 1497–1514, 1980.

[9] W.D. Harvey and M.L. Ginsberg. Limited discrepancy search. In *Proceedings of IJCAI95*, pages 607–613, 1995.

[10] Ilog. *Ilog Solver User's Manual, Version 3.0*, 1994. ILOG SA, 9, rue de Verdun, BP 85, F-94253 Gentilly CEDEX, France.

[11] Antoon Kolen and Stan van Hoesel. A constraint satisfaction approach for the radio link frequency assignment problem. Technical report, University of Limburg, 1996.

[12] R. J. Pennotti and R. R. Boorstyn. Channel assignment for cellular mobile telecommunication systems. In *Proc. National Telecommunications Conf., Dallas*, pages 16.5–1–16.5–5, 1976.

[13] C. Schulte, G. Smolka, and J. Würtz. Encapsulated search and constraint programming in Oz. In *Second Workshop on Principles and Practice of Constraint Programming*, Lecture Notes in Computer Science, vol. 874. Springer, 1994.

[14] Christian Schulte and Gert Smolka. Encapsulated search in higher-order concurrent constraint programming. In *Logic Programming: Proceedings of the 1994 International Symposium*, pages 505–520. MIT-Press, 1994.

[15] S. F. Smith and C. C. Cheng. Slack based heuristics for constraint satisfaction scheduling. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 139–144, 1993.

[16] G. Smolka and R. Treinen, editors. *DFKI Oz Documentation Series*, 1994. Available on paper or via WWW from `http://ps-www.dfki.uni-sb.de/oz/`.

[17] Gert Smolka. The Oz programming model. In *Computer Science Today*, Lecture Notes in Computer Science, vol. 1000, pages 324–343. Springer-Verlag, Berlin, 1995.

[18] Sergey Tiourine, Cor Hurkens, and Jan Karel Lenstra. Combinatorial lower bounds for the rlfap. Technical report, T.U. Eindhoven and Delft RLFAP Groups, 1996.

[19] J.A. Zoellner and C.L. Beall. A breakthrough in spectrum conserving frequency assignment technology. In *IEEE Trans. on Electromagnetic Compatibility*, volume EMC–19(3), pages 313–319, 1977.

APPENDIX – Problems FAP2 and FAP5

A FAP2

```
%%
%%  FAP2 -- completely random
%%  Parameters: <cells> <maxdemand> <mindemand> <maxinterference>
%%              <addclique> <constrainedness>
%%  {RandomFap 25 10 3 2 0.8 Distances Demand}
%%

NbCells   = 25
Demand    = demand(9 10 7 8 5 6 3 4 9 10 7 8 5 6 3 4 9 10 7 8 5 6 3 4 9)
Distances = o(o(16 1 1 2 1 1 1 2 0 1 1 1 1 2 1 2 0 2 0 1 2 1 2 2 2)
              o(1 16 0 2 0 1 1 2 1 1 1 0 1 2 1 2 2 0 2 1 2 0 2 2 2)
              o(1 0 16 0 2 1 0 0 1 1 0 2 1 2 1 0 2 1 2 1 2 2 2 2 2)
              o(2 2 0 16 0 1 2 0 1 1 2 2 0 2 1 1 2 0 0 0 2 2 2 0 0)
              o(1 0 2 0 16 1 2 2 1 1 0 2 2 2 1 1 2 2 0 2 2 2 2 1 1)
              o(1 1 1 1 1 16 2 2 1 1 1 2 2 2 1 1 2 2 0 2 2 0 2 1 1)
              o(1 1 0 2 2 2 16 2 1 0 1 2 2 2 1 1 2 2 1 2 2 1 2 1 1)
              o(2 2 0 0 2 2 2 16 1 0 1 2 2 2 1 0 2 0 1 2 2 1 2 1 0)
              o(0 1 1 1 1 1 1 1 16 1 1 2 2 2 1 2 2 1 1 2 2 1 0 1 2)
              o(1 1 1 1 1 1 0 0 1 16 1 0 2 0 1 2 2 1 0 2 2 1 0 1 2)
              o(1 1 0 2 0 1 1 1 1 1 16 1 2 1 1 2 2 1 0 0 2 1 2 1 0)
              o(1 0 2 2 2 2 2 2 2 0 1 16 2 1 1 2 2 1 1 1 0 0 0 1 1)
              o(1 1 1 0 2 2 2 2 2 2 2 2 16 1 1 2 0 1 1 1 1 2 1 1 1)
              o(2 2 2 2 2 2 2 2 2 0 1 1 1 16 0 0 1 1 1 1 1 0 1 1 1)
              o(1 1 1 1 1 1 1 1 1 1 1 1 1 0 16 1 1 1 1 1 1 1 1 1 1)
              o(2 2 0 1 1 1 1 0 2 2 2 2 2 0 1 16 0 1 1 0 1 1 1 1 1)
              o(0 2 2 2 2 2 2 2 2 2 2 2 0 1 1 0 16 1 1 0 1 0 1 1 0)
              o(2 0 1 0 2 2 2 0 1 1 1 1 1 1 1 1 1 16 1 0 1 2 0 1 2)
              o(0 2 2 0 0 0 1 1 1 0 0 1 1 1 1 1 1 1 16 2 1 2 2 0 2)
              o(1 1 1 0 2 2 2 2 2 2 0 1 1 1 1 0 0 0 2 16 1 2 2 0 2)
              o(2 2 2 2 2 2 2 2 2 2 2 0 1 1 1 1 1 1 1 1 16 2 2 1 0)
              o(1 0 2 2 2 0 1 1 1 1 1 0 2 0 1 1 0 2 2 2 2 16 2 0 0)
              o(2 2 2 2 2 2 2 2 0 0 2 0 1 1 1 1 1 0 2 2 2 2 16 2 2)
              o(2 2 2 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 2 16 2)
              o(2 2 2 0 1 1 1 0 2 2 0 1 1 1 1 1 0 2 2 2 0 0 2 2 16))
```

B FAP5

```
%%
%%  FAP5 -- Random with additional 7-clique
%%  {RandomFap 25 19 3 2 7 0.8 Distances Demand}
%%
NbCells   = 25
Demand    = demand(14 14 3 3 13 12 7 3 5 3 7 16 11 8 4 10 18 6 7 5 13 9 16 9 6)
Distances = o(o(16 2 1 1 2 2 1 0 2 1 2 0 1 1 2 1 2 2 2 1 2 1 2 2 1)
              o(2 16 2 2 1 1 2 1 2 0 2 0 1 1 2 0 2 2 2 1 2 0 2 2 1)
              o(1 2 16 1 2 2 1 1 2 2 0 1 1 1 2 0 2 2 2 1 2 2 2 2 1)
              o(1 2 1 16 1 1 2 1 2 0 1 1 0 1 2 1 2 0 2 1 2 2 2 2 1)
              o(2 1 2 1 16 2 1 1 2 1 1 1 2 1 0 1 2 1 2 1 2 0 2 2 1)
              o(2 1 2 1 2 16 2 1 2 1 1 1 0 1 1 1 0 0 2 1 0 1 2 0 1)
              o(1 2 1 2 1 2 16 0 0 1 1 1 0 1 1 1 1 2 2 1 0 1 0 1 1)
              o(0 1 1 1 1 1 0 16 1 1 0 1 2 1 0 1 1 0 2 0 0 1 1 1 1)
              o(2 2 2 2 2 2 0 1 16 0 0 1 2 1 2 1 1 1 2 2 1 1 0 1 0)
              o(1 0 2 0 1 1 1 1 0 16 1 1 2 1 2 1 0 1 2 2 1 1 2 1 0)
              o(2 2 0 1 1 1 1 0 0 1 16 1 2 1 2 1 2 1 2 2 1 1 2 1 0)
              o(0 0 1 1 1 1 1 1 1 1 1 16 0 1 0 1 2 1 2 2 1 1 2 1 2)
              o(1 1 1 0 2 0 0 2 2 2 2 0 16 0 1 0 0 0 2 2 1 1 2 0 2)
              o(1 1 1 1 1 1 1 1 1 1 1 1 0 16 1 2 1 2 2 2 1 1 2 2 2)
              o(2 2 2 2 0 1 1 0 2 2 2 0 1 1 16 2 1 0 2 2 1 0 2 2 2)
              o(1 0 0 1 1 1 1 1 1 1 1 1 0 2 2 16 0 0 2 2 1 2 2 0 2)
              o(2 2 2 2 2 0 1 1 1 0 2 2 0 1 1 0 16 2 2 2 1 2 0 1 2)
              o(2 2 2 0 1 0 2 0 1 1 1 1 0 2 0 0 2 16 0 2 1 2 1 1 0)
              o(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 16 2 1 2 0 0 1)
              o(1 1 1 1 1 1 1 0 2 2 2 2 2 2 2 2 2 2 2 16 1 2 2 2 1)
              o(2 2 2 2 2 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 16 2 2 0 1)
              o(1 0 2 2 0 1 1 1 1 1 1 1 1 1 0 2 2 2 2 2 2 16 2 1 1)
              o(2 2 2 2 2 2 0 1 0 2 2 2 2 2 2 2 0 1 0 2 2 2 16 1 1)
              o(2 2 2 2 2 0 1 1 1 1 1 1 0 2 2 0 1 1 0 2 0 1 1 16 1)
              o(1 1 1 1 1 1 1 1 0 0 0 2 2 2 2 2 2 0 1 1 1 1 1 1 16))
```