Anytime Frequency Allocation with Soft Constraints

Mats Carlsson, Greger Ottosson
{mc,greger}@csd.uu.se
Computing Science Dept., Uppsala University
PO Box 311, S-751 05 UPPSALA, Sweden

SICS, PO Box 1263, S-164 28 KISTA, Sweden

October 25, 1996

Abstract

As is well known, the frequency allocation problem for mobile telephone networks can be approximated by a graph coloring problem where vertices model transmitters, colors model frequencies, and an edge corresponds to a pair of transmitters that must not use the same frequency. However, the graph coloring analogy doesn't suffice for modeling real world problems, which involve soft constraints and general distance constraints. Thus, general constraint satisfaction techniques are called for.

In this paper, we model the frequency allocation problem as a partial constraint satisfaction problem, and give a branch & bound algorithm for solving it. We review a number of constraint solving techniques used in this problem and discuss their relevance to performance.

Keywords: anytime algorithm, frequency allocation, partial constraint satisfaction problem, global constraints.

1 Introduction

Since the '60s, there has been a lot of interest in automatic frequency allocation for mobile telephone networks, but few successful implementations have been reported. Hale's article [9] contains an excellent overview and bibliography of the early attempts.

It has been noted by many authors [2, 7, 9, 11, 13, 14] that the frequency allocation problem amounts to a graph coloring problem where vertices model transmitters, colors model frequencies, and an edge corresponds to a pair of transmitters that must not use the same frequency. Thus, a host of graph coloring methods have been used previously for solving such problems.

However, the graph coloring analogy doesn't suffice for modeling the full problem, which involves soft constraints and more general distance constraints than disequations. Thus, general constraint satisfaction techniques are called for.

An algorithm for automatic allocation of frequency groups was reported in [4], and the effectiveness of constraint lifting and other techniques was evaluated. This algorithm has since been refined to reason about individual frequencies, extended with a number of new techniques not reported in [4], and packaged into a fielded application, Freplan.

In this paper, we model the frequency allocation problem as a partial constraint satisfaction problem, and give a branch & bound algorithm for solving it. We review a number of constraint solving techniques used in this problem and discuss their relevance to performance.

2 Problem statement

A mobile telephone network is partitioned into *cells*. Each cell is served by a unique *base station*. The problem consists in assigning a set of frequencies to the cells, while minimizing the potential interference. The number of frequencies assigned to each cell varies with the required capacities of the base stations, and at least one frequency per cell must be a so called *control frequency*. Some frequencies may be preassigned to some cells; others may be forbidden. A minimal separation among frequencies within the same cell is prescribed.

The problem is modeled as a partial constraint satisfaction problem (PCSP) [5] with

variables: f_{ik}, the frequencies to be assigned to each cell i with capacity c_i, $1 \leq k \leq c_i$.

domains: The set of available frequencies. A subset of these are available as control frequencies.

constraints: There are two principal types of constraints:

- **soft inter-cell constraints:** for some pairs of cells i and j, there is an integer cost $h(i,j,d)$ if frequencies with a distance less than d are assigned to i and j. That is, if $\exists k,l : |f_{ik} - f_{jl}| < d$.
- **hard co-cell constraints:** a minimal separation m between frequencies assigned to the same cell is prescribed. That is, $\forall i,k,l : k \neq l \rightarrow |f_{ik} - f_{il}| \geq m$. Furthermore, at least one of the frequencies assigned to a cell must be a control frequency.

cost function: All co-cell constraints must be satisfied and all variables must be assigned in a valid solution. The cost of a solution is the maximal cost of any unsatisfied inter-cell constraint. This cost is to be minimized.

Input data include a list of the available frequencies, a list of the cells, their required capacities, any preassigned or forbidden frequencies, the minimal co-cell frequency separation m, and a list of inter-cell costs $h(a,b,d)$. Typical cell capacities vary between 1 and 6. The number of available frequencies is typically between 25 and 50. There are often several hundred cells, and in principle there can be several soft constraints on each pair of cells.

The cell structure is a key property of this problem. Treating the inter-cell constraints as constraints on pairs of individual frequencies, rather than on cells as a whole, would be extremely space consuming. Furthermore, the search space can be greatly reduced by observing that all permutations of frequencies assigned to a given cell are equivalent, i.e. the co-cell constraints can be strengthened to $\forall i,k : 1 \leq k < c_i \rightarrow f_{i,k+1} \geq f_{i,k} + m$.

Thus, working with constraints on whole cells is essential to making the problem tractable. This observation was first made by Pennotti [11] who introduced the concepts *micrograph* (constraints on frequency pairs) and *macrograph* (constraints on cell pairs).

Typically, the soft constraints are derived from geographic data, electromagnetic propagation data (measured or simulated), heuristic rules, or a combination of these. The details are outside the scope of this paper.

3 Basic algorithm

We use an approach reminiscent of branch & bound search to solve the minimization problem. The basic idea is to transform the PCSP to a sequence of standard CSPs and search for a CSP that is as tightly constrained as possible but still solvable.

Throughout, we maintain two bounds: α, the highest cost at which the problem is known to be unsolvable, and β, the cost of the best solution so far, and perform a binary search between the bounds. At each step in the search, we partition the soft constraints into two sets wrt. a threshold value $\theta = \lfloor(\alpha + \beta)/2\rfloor$. We then generate and solve a standard CSP consisting of the hard constraints and those soft constraints that have a cost greater than the threshold. If a solution is found, we set $\beta \leftarrow \theta$; otherwise, we set $\alpha \leftarrow \theta$. This is iterated until $\alpha = \beta - 1$.

The problem with this approach is the exponential time that may be required to solve each CSP. Freplan really requires an "anytime algorithm" that finds a reasonable solution quickly, and produces better and better solutions with increasing CPU time. To this end, the binary search has an extra parameter: a limit on the number of backtrack steps allowed in each CSP. This backtrack limit is incremented in an outer loop. Thus, an attempt to solve a CSP may terminate with success, with failure, or because the backtrack limit has been reached. In the latter case, we can't update α as a solution may exist for thresholds $\leq \theta$. Instead, θ is set to $\lfloor(\theta + \beta)/2\rfloor$.

The motivation behind the backtrack limit is to make the algorithm spend more efforts on solving CSPs in the loosely constrained region close to β, as such CSPs are likely to be solvable with a smaller number of backtracks. We are thus less likely to get stuck, and will produce better solutions incrementally, albeit not regularly.

The complete algorithm is as follows. Let b be the current backtrack limit. Initial α and β are precomputed using graph-theoretic techniques (clique detection and graph reduction).

1. Set $b \leftarrow 1$.

2. If $\alpha = \beta - 1$, terminate with a proof of optimality. Otherwise, set $\theta \leftarrow \lfloor(\alpha + \beta)/2\rfloor$.

3. Generate and solve a CSP wrt. θ and b. Three outcomes are possible:

   success: A new best solution has been found. Report this fact, set $\beta \leftarrow \theta$, and goto step 2.

   failure: No solution can exist for thresholds $\leq \theta$. Set $\alpha \leftarrow \theta$, and goto step 2.

   backtrack limit reached: A solution may exist for thresholds $\leq \theta$ but requires more than b backtracks. If $\theta = \beta - 1$, set $b \leftarrow 2b$ and goto step 2. Otherwise, set $\theta \leftarrow \lfloor(\theta + \beta)/2\rfloor$, and goto step 3.

While solving the CSP, each constraint maintains arc-consistency over its variables. Both types of constraints (co-cell and inter-cell) are *global* [1] over the variables involved, have special-purpose algorithms for maintaining consistency of their domains, and are invoked whenever one of these domains is pruned. Entailment of inter-cell constraints is detected incrementally—whenever such a constraint is detected as entailed, it is effectively removed from the constraint graph, which may consequently affect the variable choice heuristic (see Section 4.1) and improve problem reduction (see Section 4.4).

4 Optimizations and heuristics

The basic algorithm has been extended to include a number of optimizations and search heuristics.

4.1 Variable choice heuristic

The most constrained heuristic is used, i.e. a variable with the smallest domain is selected, breaking ties by selecting the variable that has the most constraints attached to it.

To support this heuristic, we maintain the cells in a priority queue indexed on pairs (s_i, d_i) where s_i is the size of the smallest domain of any unassigned variable of the cell i, and d_i is the number of active inter-cell constraints attached to i.

4.2 Value choice heuristic

When we have selected the next variable to assign, we use the min-conflict heuristic [6] to determine the order in which to try the alternative frequencies. The intuition is that the best choice has the smallest impact on the neighborhood. This is implemented by a simple look-ahead mechanism, counting the number of domain elements in neighboring cells excluded by each choice of frequency.

4.3 Iterative broadening

Even with the above heuristics, the basic algorithm is still fundamentally a depth-first tree search, and tends to get stuck in deep branches. This is a problem, especially as the domain size is fairly large initially. Thus, the first several assignments made are never reconsidered during the backtrack search, unless the backtrack limit is very high.

To avoid this problem, we use iterative broadening [8]: for any variable, we only consider its w most promising alternative frequencies (as determined by the value choice heuristic), where w is a parameter of the search. This ensures that the search space is explored more fairly.

In our implementation, we set $w \leftarrow 2$ initially in an outer loop around the whole search and increment it until the maximal domain size is reached. In practice, the problems are usually too large for an exhaustive search to be feasible, and we have not observed $w > 2$ except on small examples or after an exceedingly long execution time. This observation underscores the importance of an accurate value choice heuristic—poor value choices made early have little chance of being corrected upon backtracking.

Instead of iterating over w in an outer loop and iterating over the backtrack limit in an inner loop, one might consider intertwining the two in a single loop, but we have not experimented with that idea.

4.4 Problem reduction

In a preprocessing phase, we compute for each cell whether it can be assigned deterministically after all other cells, and at what threshold it is safe to do so. We use a variant of Matula's smallest-last ordering [10] for this purpose.

Furthermore, during the search, we may detect that a cell i with a single unassigned variable is attached to another such cell via an inter-cell constraint, or to none at all. Since arc-consistency is being maintained, we can safely remove i from the constraint graph and defer its assignment until all other cells have been assigned. Once i has been removed, its single neighbor (if any) may become a new candidate for removal, and so on.

Thus, problem reduction is being done both before and during search. During the search, a stack of cells that can be safely assigned after all other cells is maintained. As acyclic subparts of the constraint graph are incrementally removed, new cells are pushed on the stack.

If the search is successful, we can deterministically assign frequencies to the cells in the stack. Each cell i popped from the stack is assigned frequencies that minimize the interference between i and other cells that have already been assigned. This is easily done by considering the constraints on i by decreasing cost, pruning the domains of i until they cannot be pruned any more. Thus, we here consider *all* constraints on i, including those that have a cost below the current threshold, resulting in a further decrease in the total interference.

4.5 Intelligent backtracking

The algorithm uses a form of intelligent backtracking, the purpose of which is to avoid backtracking to choice points that cannot possibly cure a detected inconsistency. The algorithm is essentially identical to that developed in [3]. To support intelligent backtracking, the implementation maintains dependency information as a bit-mask per cell. This may lead to a loss of precision compared to maintaining dependency information per variable, but we have not experimented with the latter.

4.6 Global constraints on cliques

A typical instance of our frequency allocation problem is quite dense near the optimal point, with 15%-25% of all possible inter-cell constraints above the threshold. This observation prompted us to look for cliques of inter-cell constraints and replace these by global constraints over the members of the clique.

For each CSP, we examine the constraint graph and extract such cliques, replacing the binary constraints they subsume by single `all_different` constraints denoting pairwise disequations between all variables involved. We use Régin's complete algorithm [12] for maintaining consistency. This transformation has the advantages of

1. reducing the number of constraints engaged in propagation, and
2. increasing propagation power.

The clique extraction is done using a greedy algorithm. The aim is not to find the largest clique, but rather to find a set of cliques covering as large portion of the graph as possible. By allowing overlapping cliques, we increase propagation power, but also increase propagation cost.

5 Impact of optimizations

It would be reasonable to re-implement Freplan in a CLP(FD) system with support for global constraints, and efficiency gains would be expected due to low-level support for constraint propagation. The optimizations described in Section 4 are all fairly standard techniques. However, not all of them are commonly available in general CLP(FD) solvers, and the loss of these optimizations could even outweigh the efficiency gains.

In order to gauge the importance of some optimizations, we performed a limited study on some selected problem instances. Figure 1 shows the properties of the problem instances.

Instance	#Cells	#Constraints	Density[1]	#Frequencies
A	88	3221	24%	48
B	157	3333	15%	12
C	157	3333	20%	24
D	113	2367	14%	12

Table 1: The problem instances

We compared four clones of Freplan: the full version with all optimizations, one without intelligent backtracking, one without iterative broadening and one without problem reduction. We did

[1]Density denotes the ratio of constraints above optimal threshold to the total amount of constraints.

not include the variable and value choice heuristics in the study as these are commonly available in solvers. Figures 1, 2, 3, and 4 show the performance of these clones on the benchmarks.

From these plots we draw the following conclusions:

Intelligent backtracking seems to be quite important in all our problem instances.

Iterative broadening is advantageous for one instance, but detrimental for another.

Problem reduction seems to be rather insignificant.

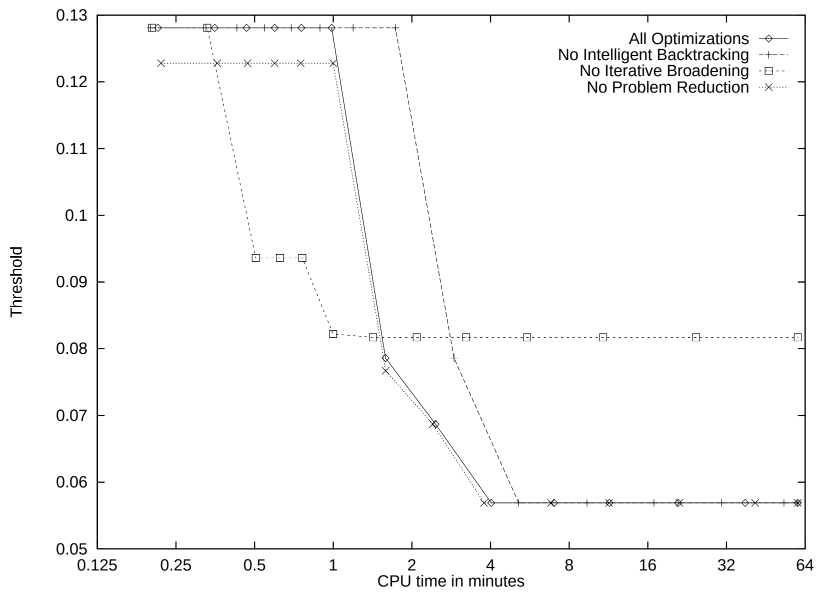

Figure 1: Problem instance A

The work on extracting cliques from the constraint graph is still in progress. Although not yet viable as an alternative when speed is considered, we show in Fig. 5 how propagation power is improved (in relation to allowed backtracking) when `all_different` constraints replace binary constraints.

6 Conclusions

We have described an application for frequency allocation in mobile telephone networks. The system is modeled using soft and hard constraints, where the goal is to minimize the maximal cost of any unsatisfied soft constraint. The minimization is done by an anytime algorithm using a branch & bound scheme. Several standard optimization techniques, in particular intelligent backtracking, are shown to be effective in this application.

We are currently investigating the idea of replacing groups of binary co-channel constraints by single n-ary `all_different` constraints. Although initial results are promising, it is not yet clear whether the stronger consistency they delivers outweighs the higher cost of the n-ary constraints.

The current version is written in plain Prolog. CLP(FD) would be an attractive alternative if it could provide support for intelligent backtracking and user-defined global constraints.

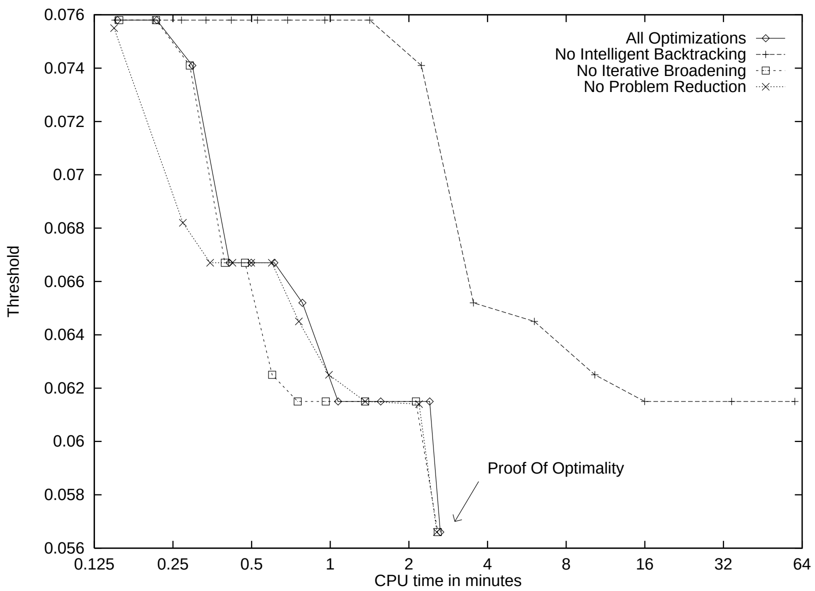

Figure 2: Problem instance B

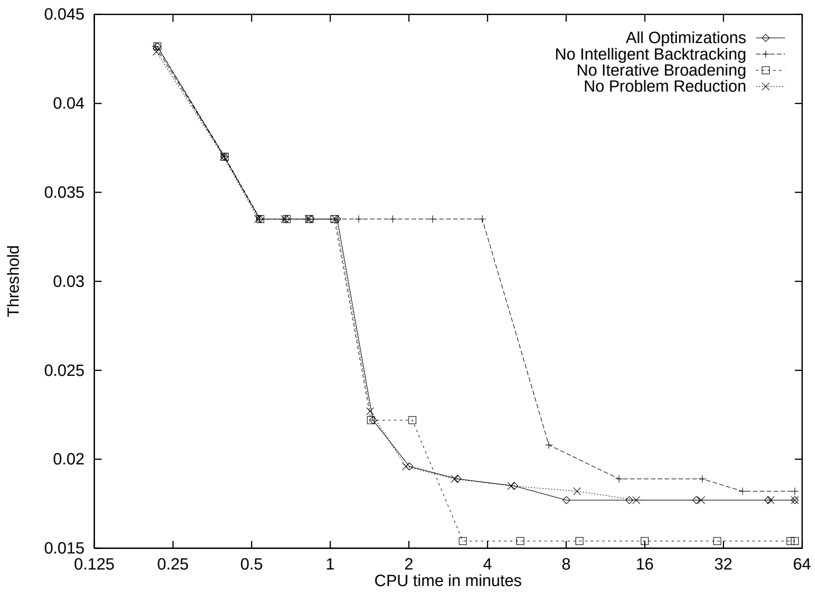

Figure 3: Problem instance C

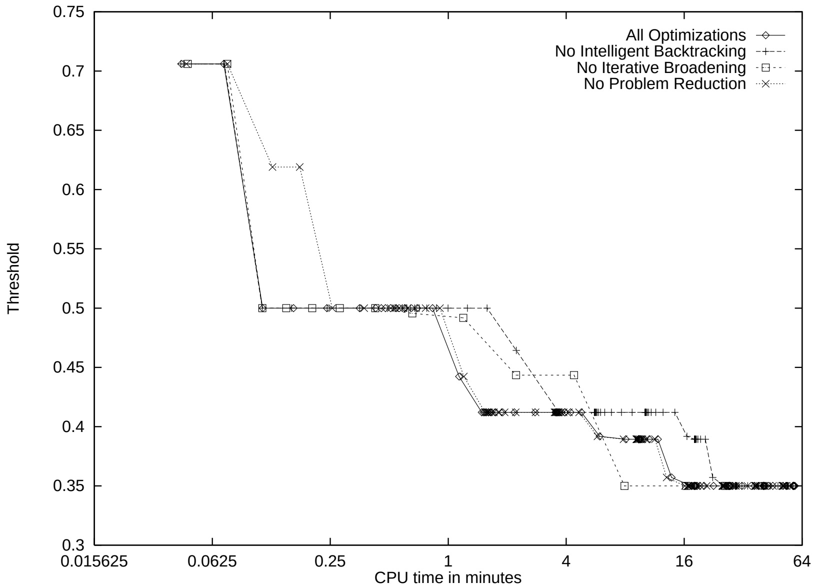

Figure 4: Problem instance D

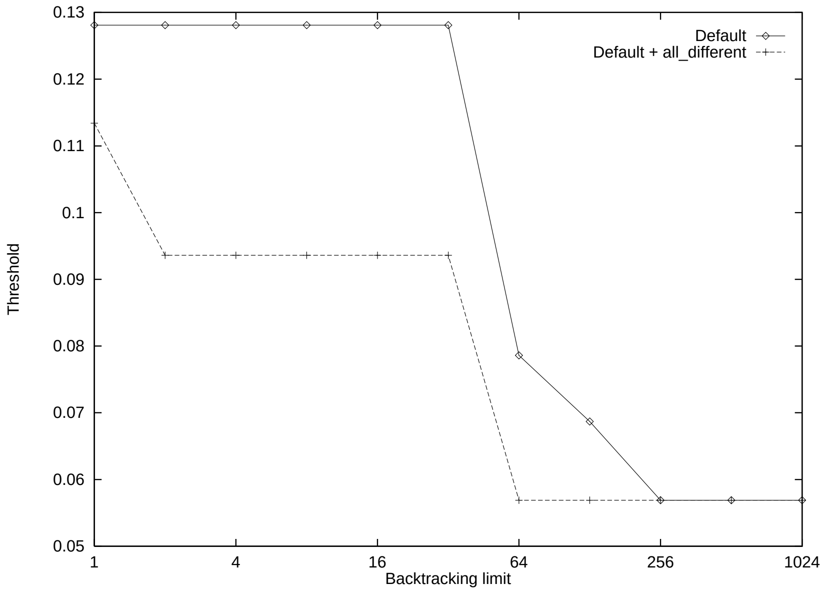

Figure 5: Propagation with `all_different` constraints

Acknowledgments

This work was supported in part by the Advanced Software Technology Center of Competence at Uppsala University (ASTEC), and in part by Ericsson Radio Systems AB (ERA). ERA introduced us to the world of automatic frequency allocation, and kindly provided the benchmark data.

References

[1] N. Beldiceanu and E. Contejean. Introducing global constraints in CHIP. *Mathl. Comput. Modelling*, 20(12):97–123, 1994.

[2] Frank Box. A Heuristic Technique for Assigning Frequencies to Mobile Radio Nets. *IEEE Trans. on Vehicular Technology*, VT-27(2):57–64, May 1978.

[3] M. Bruynooghe. Solving combinatorial search problems by intelligent backtracking. *Information Processing Letters*, 12(1):36–39, 1981.

[4] M. Carlsson and M. Grindal. Automatic frequency assignment for cellular telephones using constraint satisfaction techniques. In David S. Warren, editor, *Proceedings of the Tenth International Conference on Logic Programming*, pages 647–665, Budapest, Hungary, 1993. The MIT Press.

[5] E.C. Freuder. Partial constraint satisfaction. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 278–283, Detroit, 1989.

[6] D. Frost and R. Dechter. Look-ahead value ordering for constraint satisfaction problems. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Canada, August 1995.

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

[8] M.L. Ginsberg and W.D. Harvey. Iterative broadening. *Artificial Intelligence*, 55:367–383, 1992.

[9] William K. Hale. Frequency Assignment: Theory and Applications. *Proceedings of the IEEE*, 68(12):1497–1514, December 1980.

[10] David W. Matula and Leland L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. Technical Report CSE 8104, Department of Computer Science, Southern Methodist University, July 1981.

[11] 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.

[12] J.-C. Régin. A filtering algorithm for constraints of difference in CSPs. In *Proc. of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 362–367, 1994.

[13] K.N. Sivarajan, R.J. McEliece, and J.W. Ketchum. Channel Assignment in Cellular Radio. In *Proc. 39th IEEE Veh. Technol. Soc. Conf.*, pages 846–850. IEEE, May 1–3 1989.

[14] J.A. Zoellner and C.L. Beall. A Breakthrough in Spectrum Conserving Frequency Assignment Technology. *IEEE Trans. on Electromagnetic Compatibility*, EMC-19(3):313–319, August 1977.