A Branch-and-Cut Algorithm for the Frequency Assignment Problem

K.I. Aardal
A. Hipolito
C.P.M. van Hoesel
B. Jansen

Abstract

The frequency assignment problem (FAP) is the problem of assigning frequencies to transmission links such that no interference between signals occurs. This implies distance constraints between assigned frequencies of links. The objective is to minimize the number of used frequencies. We present an integer linear programming formulation that is closely related to the vertex packing problem. Although the size of this formulation is an order of magnitude larger than the underlying network of links, we use the integer linear programming formulation within a branch-and-cut algorithm. This algorithm employs problem specific and generic techniques such as reduction methods, primal heuristics, and branching rules to obtain optimal solutions. We report on computational experience with real-life instances.

Table of contents

1 Introduction

In mobile telephone systems communication between pairs of telephones takes place by wireless connections, which make use of frequencies from the electromagnetic band. Connections may interfere with each other whenever they are close to one another, both with respect to geographical distance, and distance of the assigned frequencies on the radio band. To reduce the effect of interference to a minimum, links with a small geographical distance, should be assigned frequencies with a high difference in value. On the other hand, the frequencies are available in limited numbers, and there are many different users, which makes it expensive to hire them. Therefore, a client would like to use a minimum number of different frequencies.

More formally we may describe the frequency assignment problem (FAP) as follows. A set of transmission links L need to be established in order to enable communication between a set of stations. Each link $i \in L$ has to be assigned a frequency f from a given available set of frequencies D_i, the domain, such that no interference occurs between given subsets of links, i.e. such that for every pair of links (i, j) the frequencies f_i and f_j assigned to these links differ by at least d_{ij}. The union of the domains is denoted by D. The objective is to minimize the number of used frequencies.

The FAP occurs in many variants. For instance, another realistic objective is to minimize the span of the frequencies chosen, i.e., the difference between the largest and smallest frequencies chosen. We shall also study a variant of FAP, in which we allow for interference which is penalized, i.e., distance constraints can be violated against a prespecified penalty. Moreover, we assign preferred frequencies to a subset of the links. If another frequency, over the preferred one is chosen to satisfy the distance constraints, then a penalty is incurred. The objective is to minimize the total penalty.

The FAP is a hard problem as it is closely related to the graph coloring problem. This is probably the reason that the majority of the papers on FAP describe heuristic procedures for finding good solutions. For a detailed overview on the latest developments with respect to heuristics and exact methods see Tiourine et al. [11]. The relation to graph coloring provides some interesting lower bounds as described in Gamst [1] and Lanfear [3]. Exact solution methods have had little attention so far, with the exception of van Hoesel et al. [2], who use combinatorial Branch and Bound with constraint satisfaction techniques. We present an algorithm for the FAP based on an integer linear programming formulation of the problem as a vertex packing problem. This formulation is used in a cutting plane algorithm, which in turn is embedded in a branch-and-cut framework. The vertex packing formulation has a disadvantage in the sense that it is an order of magnitude larger than the size of the network of stations and links: the number of variables involved is $\mathcal{O}(|L||D|)$. To cope with such sizes we incorporate specially designed separation routines for the cutting plane algorithm, and variable selection mechanisms as well as reduction techniques to keep the formulation as small as possible. It should be noted that we intend to solve vertex packing problems with up to 16000 variables, whereas previous studies using similar techniques stop at a mere 200 variables, see [8]. Jünger et al. [4] provide a survey of branch-and-cut applications, focused on implementational aspects.

The outline of the paper is as follows. We describe the formulation for the FAP in Section 2. Section 3 describes some classes of valid inequalities for the FAP, with separation routines.

It also contains a description of the implementational aspects of the cutting plane algorithm. The branch-and-cut framework and its components can be found in Section 4. Preprocessing techniques to decrease the size of the formulation are described, as well as primal heuristics to generate integer feasible solutions, and a selection of branching rules. Section 5 contains computational results on the CELAR test set, a set of real-life instances from military applications. Concluding, in Section 6, we discuss how the branch-and-cut algorithm can be applied to tackle variants of FAP.

2 Problem formulation

The FAP can be represented on a graph $G_I = (V_I, E_I)$ as follows. The set of vertices V_I of the graph correspond to the set of transmission links L. A pair of vertices $i, j \in V_I$ is connected by an edge $\{i, j\} \in E_I$ if there is a distance requirement on the frequencies f_i and f_j that are to be assigned to i and j. For each link i, there is a set of admissible frequencies D_i, the *domain* from which the frequency for i must be chosen. The union of all domains is denoted by D.

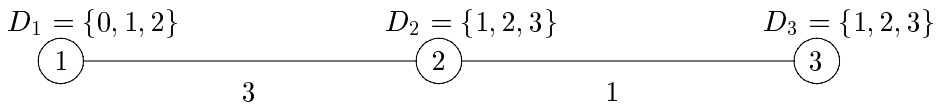
We refer to the graph $G_I = (V_I, E_I)$ as the *interference graph*.

The distance requirement on the frequencies are

$$|f_i - f_j| \geq d_{ij} \qquad \text{for all } \{i, j\} \in E_I \tag{1}$$

where $d_{ij} > 0$.

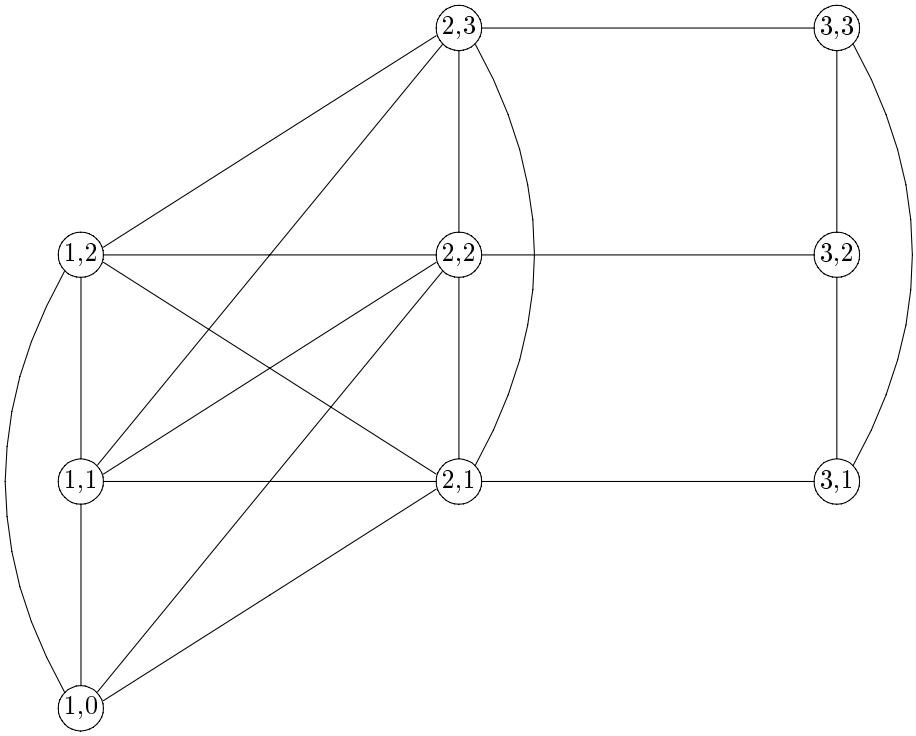
Example



FAP instance.

The above problem representation is not a convenient one for integer programming. An alternative representation for the FAP, which leads more naturally to a mathematical programming formulation, is given by an extended graph $G_X = (V_X, E_X)$. We introduce a vertex $v_{if} \in V_X$ for each combination of a link i and a frequency $f \in D_i$, An edge between two vertices v_{if} and v_{jg} corresponds to a violation of the distance constraint between i and j, i.e., $|f - g| < d_{ij}$. We call G_X the *graph of variables*. A feasible solution for the FAP consists of a subset S of V_X, such that exactly one vertex corresponding to each link $i \in L$ is chosen S. Moreover, S should form a vertex packing in G_X, i.e., no two vertices in S are allowed to be connected by an edge. The graph G_X for the above example is as follows.

Example (continued)



FAP instance: graph G_X.

The objective is to find a set S which uses the minimum number of frequencies, i.e., the set of frequencies corresponding to vertices in S must have minimum cardinality. The corresponding integer programming formulation, FAPVP, is given below.

$$x_{if} = \begin{cases} 1 & \text{if frequency } f \text{ is assigned to link } i \\ 0 & \text{otherwise,} \end{cases}$$

$$y_f = \begin{cases} 1 & \text{if frequency } f \text{ is used} \\ 0 & \text{otherwise.} \end{cases}$$

$$\text{(FAPVP)} \quad \min \sum_{f \in D} y_f \tag{2}$$

$$\text{s.t.} \quad \sum_{f \in D_i} x_{if} = 1 \qquad \text{for all } i \in L \tag{3}$$

$$x_{if} + x_{jg} \leq 1 \qquad \text{for all } i, j \in L : \ |f - g| < d_{ij} \tag{4}$$

$$x_{if} \leq y_f \qquad \text{for all } i \in L, f \in D_i \tag{5}$$

$$x_{if}, \in \{0, 1\} \qquad \text{for all } i \in L, f \in D_i \tag{6}$$

$$y_f \in \{0, 1\} \qquad \text{for all } f \in D \tag{7}$$

Note that the constraints 5 can be converted to vertex packing constraints, by using the complement of y_f ($f \in D$), i.e., by taking $\bar{y}_f = 1 - y_f$. This substitution will turn 5 into

$$x_{if} + \bar{y}_f \leq 1 \qquad \text{for all } i \in L, f \in D_i \tag{8}$$

Clearly, one can extend G_X by adding nodes for the frequencies, which correspond to the variables $\bar{y}_f$ ($f \in D$).

Since the cardinality of each domain D_i tends to be large this formulation contains many variables. This problem is particularly important when solving the FAP with branch-and-cut. In the following sections we will show how to deal with this problem in the various components of the algorithm. For now, we will restrict ourselves to the positive side of the extended formulation. If the FAP is relaxed by setting the distances $d_{ij} = 1$ for each edge $\{i, j\} \in E_I$, we obtain the *generalized coloring* relaxation, also known as T-coloring problem. Relaxing even more by setting $D_i = D$ ($\forall_{i \in L}$) we get the ordinary *coloring* relaxation. These two relaxations provide lower bounds on the solution value of FAP. An even weaker, but easier to determine, lower bound is the size of a maximum clique in G_I. The clique bound is outperformed by the bound obtained by the linear programming relaxation of FAPVP where additional clique inequalities are added. If the distances play an important role, this bound may also be better than the (generalized) coloring bound. In the example above addition of clique inequalities leads to a lower bound of 3: link 1 must be assigned frequency 0, and link 2 must be assigned frequency 3; finally link 3 must be assigned one of the frequencies 1 and 2. The combinatorial lower bounds all have value 2. Note that it is often hard to compute the (generalized) coloring bound.

In practice two-way traffic through a link i is realized by assigning two frequencies to i, one for each traffic direction. These frequencies should have a constant difference δ for all links in the network. In many practical situations we can forget about this situation. The reason for this is that the distance requirements for both frequencies are equal and the domain D is partitioned into two parts which are far enough to ensure that all distance requirements between links that have frequencies in different parts are met. A solution in which one of the directions is assigned a frequency in the first part can then be copied for the second part. However, in our case we have no symmetry in the distance requirements, nor do we have domains which can be partitioned into two distant bands. Instead we have pairs of opposite directions of a link, that must be assigned a pair of frequencies, such that for both directions all the distance requirements are satisfied. The frequencies come in pairs too, i.e., for each frequency f there is a unique frequency f' with the property that $|f - f'| = \delta$ In the sequel we will view the directions as separate links. Moreover, we will denote the opposite link of a link i by i', and the frequency paired to f by f'.

In the graph of variables each vertex v_{if} corresponds to a unique vertex $v_{i'f'}$, where i and i' are opposite traffic links, and f' is the unique frequency such that $|f - f'| = \delta$. An edge between two such vertices models an equality constraint, i.e., $x_{if} = x_{i'f'}$. This equation has implications for the edge-set in the graph of variables. If a distance constraint forbids the combinations (i, f) and (j, g), then also the combinations of the corresponding opposite links (i', f') and (j', g') are forbidden. Thus, we may draw an edge between the corresponding variables as well. Note that this extension can not always be made in the interference graph: if two links i and j must have distance at least d_{ij}, this does not necessarily apply to the opposite links. For example, take $D_i = D_{i'} = D_j = D_{j'} = \{1, 2, 3, 7, 8, 9, 11, 12, 13, 17, 18, 19\}$ and set $\delta = 6$. If $d_{ij} = 4$, then $f_i = 3$ and $f_j = 17$ are valid choices. However, the opposite

links i' and j' have frequencies 9 and 11, which do not satisfy the distance requirement for i and j. In the sequel we shall make implicit use of this special structure at several occasions: in the construction of valid inequalities, in preprocessing ideas, and in the branch-and-cut framework.

3 The cutting plane algorithm

The linear relaxation LRFAPVP of FAPVP is weak, which is a well-known fact for vertex packing related problems, see Padberg [9]. Therefore, LRFAPVP must be strengthened with strong valid inequalities. Since the vertex packing problem is a relaxation of FAPVP (relax 3 to ≤ 1), constraints that are valid for the vertex packing polytope are also valid for FAPVP. There are many classes of valid inequalities known for the vertex packing problem, among which the best known ones are the clique inequalities and the (lifted) odd-hole inequalities, see for instance Padberg [9]. Odd-hole inequalities and many other classes do not perform well for vertex packing problems where the average value of the variables is small. In FAPVP the average value is $\frac{1}{|D_i|}$ for each link $i \in L$. In all types of instances that we have encountered the average domain sizes are 30 or more. Concluding, we restrict ourselves to inequalities derived from cliques only in G_X.

Cliques in the graph of variables G_X Let the set $C \subset V_X$ form a clique in the graph of variables. The corresponding clique inequality is

$$\sum_{v_j \in C} x_j \leq 1 \tag{9}$$

A clique inequality defines a facet of the convex hull of feasible solutions of the vertex packing relaxation if C is maximal (see e.g. Padberg [9]). The clique inequalities for FAPVP form the basic ingredients of the cutting plane algorithm. Nevertheless, we take only a subset of these inequalities. The reason for that is the size of the graph of variables. We consider instances with up to 916 links with average domain size of about 40. The graph of variables would consist of approximately 36000 paired vertices. Moreover, the number of edges in this graph is so large that it is senseless to construct it explicitly. Finally, finding large cliques in this graph is an enormous task.

Cliques in the interference graph G_I The clique inequalities we identify are special classes of clique inequalities, derived from cliques in the interference graph. Take a subset of the links $I \subset L$, and take for each link $i \in I$, a subset of the frequencies in its domain, say $F_i \subset D_i$. Let F_I be the union of the sets of frequencies of the links in I. The type of cliques we will identify are special structures of the following constraint type

$$\sum_{i \in I} \sum_{f \in F_i} x_{if} \leq 1 \tag{10}$$

In the special case that F consists of only one frequency f the right-hand side of (10) may be reduced as follows if I is a clique in the interference graph:

$$\sum_{i \in I} x_{if} \leq y_f \tag{11}$$

Extended cliques in the interference graph G_I In a way similar to the extension of the edge-set of the graph of variables we may, under specific conditions, be able to extend the interference graph, with additional edges, or the increase the distance of certain edges.

Consider a set of frequencies F with the property that for all $f \in F$ the opposite frequency f' is $f + \delta$ (or $f - \delta$). Then, for each edge $\{i, j\}$ with distance d_{ij} we may explicitly add the constraint $|f_{i'} - f_{j'}| \geq d_{ij}$ for the opposite links i' and j' to the interference graph, since

$$|f_{i'} - f_{j'}| = |(f_{i'} - \delta) - (f_{j'} - \delta)| = |f_i - f_j| \geq d_{ij}$$

In this extended interference graph we look for cliques, in which all edges have a distance of at least d. Let C be such a clique, and let F be a set of frequencies such that the largest distance between any pair of frequencies in F is smaller than d. Then the following inequality is valid:

$$\sum_{i \in I} \sum_{f \in D_i \cap F} x_{if} \leq 1 \tag{12}$$

Take for example $D_i = D_{i'} = D_j = D_{j'} = \{1, 2, 3, 7, 8, 9, 11, 12, 13, 17, 18, 19\}$ and set $\delta = 6$. If $d_{ij} = 3$, then for $I = \{1, 2, 3\}$ we may add the distance requirement $d_{i'j'} = 3$.

In case F consists of only one frequency, we have only one opposite frequency, thus the condition on F is always fullfilled, and therefore the corresponding inequality (11) is also valid for cliques in the extended interference graph.

Separation The problem of finding violated clique inequalities has now been reduced to finding cliques in the (extended) interference graph. Since the size of these cliques is not too large, this is a plausible idea. In our instances we have another advantage of looking for the above type of inequalities in the fact that for a specific clique in the interference graph we can find many sets F satisfying the requirements for which (11) and (12) are valid. Therefore, we treat each clique in the interference graph as a basis for a whole class of inequalities.

KAREN : Hier iets over de pool?

4 The branch-and-cut algorithm

We have developed a branch-and-cut algorithm for the FAP based on the FAPVP formulation in section 2. The algorithm has the following components besides the cutting plane algorithm.

- Preprocessing to reduce the size of the problem formulation. This is highly effective for the initial problem, and is often useful in intermediate stages.
- Instance reduction techniques for the interference graph. This incorporates the need for extension heuristics.
- Upper bounding. Heuristic methods based on the linear programming relaxation are capable of finding good solutions. In combination with the lower bounds provided by the cutting plane algorithm the search process is substantially reduced.

- Branching rules for creating subproblems and selection of subproblems form the basis of the search process in the framework. The search tree in a branch-and-cut framework contains a large list of subproblems. Maintenance of the linear constraints is important.

Each of these components is discussed in the following subsections.

4.1 Preprocessing: Reducing the Problem Size

For expository reasons we maintained copies of variables in the graph of variables, i.e., each variable x_{if} has the same value as the variable $x_{i',f'}$ where i' is the opposite link of i and f' is the frequency corresponding to f: $|f - f'| = \delta$. If we substitute these variables out of the formulation this reduces the number of variables with a factor 2.

Technique 1 One way of reducing the number of x_{if}-variables, other than by mere substitution, is by trying to identify links that can always be assigned the same frequency as another link. If such a link is identified it can be deleted from the instance. In terms of graph coloring: a vertex w whose neighbor set contains the neigbor set of a vertex v can be removed if v and w are not connected. Consider a pair of links (i, j). If the following conditions are satisfied, it is always possible to assign the same frequency to link i as to link j, without affecting the objective function negatively.

Dominance criteria

(1) $D_j \subseteq D_i$,

(2) $d_{ij} = d_{i'j'} = 0$,

(3) $d_{kj} \geq d_{ki}$ for all $k \neq i, j$,

(4) $d_{k,j'} \geq d_{k,i'}$ for all $k \neq i', j'$.

Our experience shows that the number of x_{if}-variables is reduced by ten to fifteen percent. For more details see the section on computations. The problems with these variables removed are really easier, since the removed vertices often have a large degree in the interference graph.

In terms of coloring the conditions (1) to (4) can be described as follows. Let two vertices i and j have neighbor sets $N(i)$ and $N(j)$, respectively. If $N(i) \subseteq N(j)$ and $i \notin N(j)$, then we can color i with the color used for j, in any feasible solution without increasing the number of colors.

Technique 2 The second technique relies on the following idea. A link can be removed if for any choice of frequencies of the neighbors in the interference graph one can find a frequency such that the number of frequencies used is less than or equal to a given lower bound on the number of frequencies necessary for the overall problem. This technique is derived from a similar technique for the graph coloring problem, where vertices with a degree smaller than the coloring number can always be assigned a color among the ones we select to color the rest of the vertices. In the case of the frequency assignment problem, however, it also depends on the distances of the chosen frequencies. This technique reduces the number of variables with

about 5%, on the instances that we used. But more important, this idea leads to an ordering of the vertices which can be used to maintain smaller instances.

Instance reduction Before starting the branch-and-cut procedure we generate a set of cliques in the (extended) interference graph, which are stored in memory. This so–called clique–list consists of cliques that are maximal (i.e., no link can be added without decreasing the minimum distance among the vertices in the clique). Violated inequalities are found by browsing the clique–list and adding constraints that are 'sufficiently' violated (i.e., within a specified tolerance). To keep the formulation as small as possible we start-off with a subset of the links, which with a high probability contains the 'hard part' of the instance. Hereto we derive an order of the links as follows. Remove the links with the lowest degree with its incident edges, in the interference graph. Repeat this until the graph is empty. The order of removal is reversed to get our order. Our initial instance consists of the first 20 links. For this set we construct a number of clique inequalities, which form our initial formulation. This instance is solved, and then we try to extend its solution to a solution for the complete problem by a greedy algorithm in the order described before. If this procedure fails we extend the set of links with the problem link and all the links that predecess this link in the described order.

4.2 Generating primal integer feasible solutions

The availability of good feasible solutions may reduce the size of the branch-and-bound tree significantly. Furthermore, it is possible that the branch-and-cut process will be unable to prove optimality within reasonable time and it is thus important that the process be able to give good feasible assignments along the way.

We have developed a primal heuristic which tries to construct a feasible solution starting from the current fractional solution. It is based on the idea that the solution to the current linear programming relaxation can indicate good candidates for which frequencies to use and what links to assign them to.

Let the current LP solution be given by (x^*, y^*). Let F^* be the set of frequencies for which the corresponding y_f^* is positive. F^* consists of the set of candidate frequencies we will use for the computation of an assignment, the other frequencies will not be used. If a link has an already assigned frequency (by branching or forced by the LP), then this assignment is not changed anymore; if certain assignments are ruled out by branching or LP then these are also not considered in the primal heuristic. The remaining links are assigned as follows.

- For each link compute the number of frequencies that can be assigned to it, and take the link (say i) for which this number is minimal.
- For each frequency f that can be assigned to i, compute the total number of possible assignments to the remaining links that would remain if f is assigned to i; assign that frequency to i for which this number is maximal.
- Continue with the next link.

If all the links have been successfully assigned a frequency, we try to improve upon the assignment as follows. Consider the frequency that is used the minimal number of times; we try to reassign to each link that uses this frequency another frequency that is already used in the solution. Reassignment of link i is done if: (1) a frequency is found that can be assigned to i such that no interference with the other links exists, or (2) if a frequency is found that can be assigned to i such that there is one link with which it interferes and this link can also be reassigned such that no interference exists.

The quality of the upper bound of a solution found in this way may still be improved by subjecting it to other local search algorithms.

4.3 Branching Strategies

In our computational study we have implemented various branching strategies. The first one is designed to complement the primal heuristic. The second is designed to give good lower bounds on the problem.

Branching on assignment decisions The first branching rule is as follows:

- If there are fractional y_f variables, branch on the variable y_f with value closest to 1. Set $y_f = 1$ on one branch and $y_f = 0$ on the other. Evaluate the $y_f = 1$ node first. This has the effect of using the frequency that the LP relaxation indicates to be the best candidate.
- If there are no fractional y_f variables, branch on an x_{if} variable that will force the most other x variables to zero. Set $x_{if} = 1$ on one branch and $x_{if} = 0$ on the other. Evaluate the $x_{if} = 1$ node first. Note that this assignment tends to contradict the rule used for the primal heuristic. In this manner, the primal heuristic is more likely to see different measures of desirability in making assignments and perhaps increase the chance that it will identify a good feasible solution.

Branching on pairs of links In the second strategy we choose two links i, j, that are assigned a fractional amount of the same frequency and have $d_{ij} = 0$. Two subproblems are created: we enforce in the first one the constraint that links i and j are assigned the same unspecified frequency, and in the second subproblem a distance constraint $d_{i,j} = 1$ is added, i.e., i and j cannot be assigned the same frequency. Then this branching rule is illustrated by Figure 1 with an interference graph with 5 links, where connected links must be assigned different frequencies.

Observe that subsets of the vertices with many edges (subgraphs that almost form cliques) are logical candidates to use in this branching rule, since we know that using it may give a better lower bound in at least one of the branches.

Subproblem selection rules The above branching strategies specify how to partition the current set of feasible solutions into two smaller sets. They do not specify which subproblem to choose when there are several candidates. Since the first branching rule is designed to

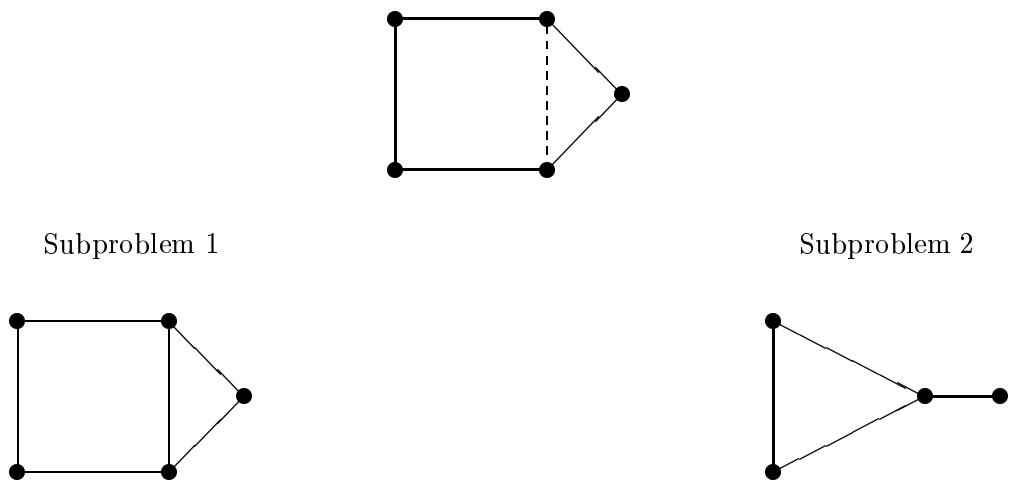


Figure 1: Branching on two links

generate good feasible solutions fast, we use the depth-first search rule on it. Best-bound search is used on the second branching rule as it is used to generate good lower bounds.

5 Computational results

The branch-and-cut algorithm has been implemented using MINTO, a Mixed INTeger Optimizer [Nemhauser, Savelsbergh, and Sigismondi 1994]. MINTO is a software system that solves mixed-integer linear programs by a branch-and-bound algorithm with linear programming relaxation. This package provides a shell within which problem-specific routines can be programmed in C and integrated into a branch-and-cut algorithm. The CPLEX 3.0 LP package was used for the LP-solving within MINTO. We used the simplex method for solving LPs. Using the steepest edge pricing rule appeared to very efficient on the problems. In our computational experience we observed it often to be better to restart solving the LP from scratch (using primal simplex and preprocessing) after adding violated inequalities instead of using the dual simplex with a warm start.

The algorithm was tested on the non-trivial feasible instances of the CELAR test set. As mentioned above, preprocessing techniques were used to reduce the size of the initial formulation. The results on preprocessing are given in Table 1. It gives the percentage reduction on the number of links used in the model. In the last column of the table one can find the CPU times (on a 486-66) for performing the reductions in seconds.

Table 2 below lists some computational results using branch-and-cut on the feasible RLFAP instances using the pre-processed formulations obtained using technique 3. These results are based on the branching rule based on assignment decisions and depth-first search. Except for CELAR05, the solution values listed are based on minimizing the number of frequencies. The values for CELAR05 are based on minimizing the largest used frequency.

The times reported are CPU times for an HP9000/720 with 144 MB of core memory, of

Instance	Size	Reduced size
1	916	806
2	200	166
3	400	348
11	680	646

Table 1: Preprocessing results.

which we used at most approximately 40MB. They do not include times for pre-processing and the generation of initial lower bounds. The table includes CELAR04 and CELAR05 for completeness although branch-and-cut was not used for them as they were trivial enough to be pre-processed to optimality.

Instance	Initial lower bound	Found lower bound	Best value	time (sec)	Approx.
1	14	16	16	400	
2	14	14	14	23	
3	14	14	**14**	539	
4	n.a.	46	46	1 (preprocessed)	
5	n.a.	792	792	2 (preprocessed)	
11	20	**22**	22	6167[1]	

[1]Lower bound of 22 verified after 413 sec.

Table 2: branch-and-cut results: feasible instances.

The branch-and-cut procedure was able to find within reasonable time, optimal solutions to all CELAR instances. More significantly, it was able to improve the previously known lower bounds of CELAR01 and CELAR11, thereby determining the optimal solutions of both in the process.

Lower bounds We compare the lower bounds given by combinatorial arguments, such as the maximum clique, coloring number and generalized coloring number with the bounds that we have obtained so far in the following table. The (generalized) coloring bounds are taken from van Hoesel [2].

Instance	clique bound	color. bound	gen. color. bound	LP-bound
1	12	14	16	16
2	14	14	14	14
3	12	14	14	14
11	20	20	20	22

Table 3: Lower bounding results: feasible instances.

6 A Variant of the Frequency Assignment Problem

One variant of FAP is to allow for some interference constraints to be violated against a penalty. The total penalty should then be minimized. Instances 6-8 are all of this type. In

instances 9-10, additional penalties are introduced for links that are not assigned a preferred frequency. The constraints that are allowed to be violated are called *soft constraints*. Let S denote the set of all pairs of links (i, j) for which the corresponding interference constraint is soft.

To be able to write the soft constraints in a linear form we introduce one variable z_{ij} for each such constraint, where

$$z_{ij} = \begin{cases} 1 & \text{if } (i,j) \in S \\ 0 & \text{otherwise.} \end{cases}$$

Let F be a set of frequencies such that the difference between any pair of frequencies in F is less than d_{ij}. The following soft constraint replaces constraint (4) wherever applicable.

$$\sum_{f \in F} x_{if} + \sum_{f \in F} x_{jf} \leq 1 + z_{i,j} \tag{13}$$

Let the penalty associated with violation of the soft constraint involving the pair (i, j) of links be equal to p_{ij}. The new objective function is given by

$$\min \quad \sum_{(i,j) \in S} p_{ij} z_{ij}.$$

To strengthen the linear relaxation of this new formulation we can use modified versions of the clique inequalities (11) and (12). Say for instance the we have a clique of 3 links, where each distance is d, and each constraint penalty is at least p. Then the following inequalities are both valid, and they tighten the linear programming relaxation.

$$\sum_{f \in F} x_{i,f} + \sum_{f \in F} x_{j,f} + \sum_{f \in F} x_{k,f} \leq 1 + z^1_{i,j,k} + z^2_{i,j,k}$$

and

$$z^1_{i,j,k} + 2 * z^2_{i,j,k} \leq z_{i,j} + z_{i,k} + z_{j,k}.$$

The variables $z^1_{i,j,k}$ and $z^2_{i,j,k}$ are both binary. Their sum determines the number of variables, that are assigned value 1, over the one that is allowed to have value 1 in the left-hand side of this inequality.

This inequality can easily be generalized to obtain valid inequalities for larger cliques.

We have made preliminary computations based on these ideas and unfortunately, the resulting lower bounds are still very poor although they prove that the tested problems are indeed infeasible (Table 4.). For CELAR07 we obtained only poor upper-bounding solutions, since our strategies so far were just attuned to improving lower bounds instead of finding good solutions. The time required to generate the lower bounds and best value is approximately 30 minutes.

7 Conclusions

The main results indicate that the branch-and-cut framework can be a powerful approach in dealing with feasible instances of the RLFAP in terms of

Instance	Found lower bound	Best value
6	5	3942
7	5	

Table 4: branch-and-cut results: infeasible instances.

- generating optimal solutions (CELAR01,02,03,11)
- improving existing lower bounds (GRAPH??), and
- generating good primal feasible solutions (GRAPH??)

within reasonable computational effort.

In the case of CELAR01 proving optimality is still a difficulty. Nevertheless, the branch-and-cut process is able to find good primal solutions. Generating even tighter cuts and/or devising new branching rules that might improve the generated lower bounds further is an area that can be further explored.

Clearly, much work still need to be done before the infeasible models can be tackled effectively using the branch-and-cut framework. Further research can be done on coming up with a more compact formulation as well as more effective pre-processing techniques and cuts for lower bounding. In addition, it may be possible to integrate the successful heuristic methods of other groups in generating primal low-violation solutions in a branch-and-cut framework. Overall, the results of this work element indicate that the branch and cut framework is flexible enough to allow the incorporation of a wide variety of techniques (from heuristics to exact methods) as well as problem-specific information in dealing effectively with radio link frequency assignment problems.

Acknowledgments

We would like to thank Martin Savelsbergh for providing helpful pointers on using MINTO and Ed Klotz and Irv Lustig of CPLEX Optimization for their timely technical support for CPLEX.

References

[1] A. Gamst, "Some Lower Bounds for a Class of Frequency Assignment Problems", IEEE Trans. on Vehicular Technology, vol. VT-35, no. 1, 8-14, 1986.

[2] C. van Hoesel, and A. Kolen, and R. van de Wal, "Constraint satisfaction techniques for the frequency assignment problem", to appear, 1996.

[3] T.A. Lanfear, "Graph theory and radio frequency assignment", Allied Radio Frequency Agency, NATO Headquarters, Brussels, 1989.

[4] M. Jünger, G. Reinelt, and S. Thienel, "Practical problem solving with cutting plane algorithms for combinatorial optimization," Report No. 94.156, Institut für Informatik, Universität zu Köln, 1994.

[5] M. Jünger, G. Reinelt, and S. Thienel, "Provably good solutions for the traveling salesman problem," Preprint 92-31, Interdisziplinäres Zentrum für Wissenschaftliches Rechnen, Universität Heidelberg, 1992.

[6] G.L. Nemhauser, M.W.P. Savelsbergh, and G.S. Sigismondi, "MINTO, a Mixed INTeger Optimizer," *OR Letters* **15** 47-58, 1994.

[7] G.L. Nemhauser, and M.W.P. Savelsbergh, "A cutting plane algorithm for the single machine problem with release times," M. Akgul, H. Hamacher, S. Tufekci (eds.) *Combinatorial Optimization: New Frontiers in the Theory and Practice*, NATO ASI Series F: Computer and Systems Sciences **82** 63-84, Springer-Verlag, 1992.

[8] G.L. Nemhauser and G. Sigismondi, "A strong cutting plane/branch and bound algorithm for node packing," *J. Opl. Res. Soc.* **25** 443-457, 1992.

[9] M.W. Padberg, "On the facial structure of set packing polyhedra," *Mathematical Programming* **5** 199-215, 1973.

[10] M.W.P. Savelsbergh, "A branch-and-price algorithm for the generalized assignment problem," Report COC-93-02, Georgia Institute of Technology, 1993.

[11] S. Tiourine, and C. Hurkens, and J.K. Lenstra, "An overview of algorithmic approaches to frequency assignment problems," COSOR Memorandum, Eindhoven University of Technology, 1995.