Lower bounds for Minimum Interference Frequency Assignment Problems

Arie M.C.A. Koster\textsuperscript{1,2}  Stan P.M. van Hoesel\textsuperscript{1}  Antoon W.J. Kolen\textsuperscript{1}

October 28, 1999

Abstract

In this paper we describe a new lower bound procedure for the minimum interference frequency assignment problem (MIFAP). In the MIFAP we have to assign frequencies to transmitter-receiver pairs in such a way that the cumulative interference in the communication network is minimized. Mainly due to the fairly large number of available frequencies, exact methods fail to solve the more difficult instances. In this paper we describe a procedure that produces a non-decreasing sequence of lower bounds. In each iteration of the algorithm we have to solve MIFAPs that are substantially smaller than the original instance. These subproblems can be solved with either integer programming techniques or a dynamic programming algorithm based on a tree decomposition of the underlying graph. Computational results show that integer programming and dynamic programming supply very good results.

1 Introduction

In the last decade the use of wireless communication technology has been increased tremendously. Not only cellular phones have become very popular but wireless communication is also used as a supplement to the capacity of existing wired connections. A wireless communication connection operates on a frequency from the radio spectrum (or two frequencies in case of bidirectional traffic). The transmitter of a connection modulates the frequency. The receiver detects these modulations, and translates them to voice or data. In a wireless communication network we have to assign frequencies to multiple connections at the same time. Depending on (among other things) the geographic locations of the connections, the use of a single frequency for two connections may cause an unacceptable level of interference (i.e., interference that exceeds the signal-to-noise ratio). Also close frequencies can cause unacceptable interference. On the other hand, we have to use a single frequency multiple times, since the available radio spectrum for wireless communication is limited. The assignment of available frequencies to connections in such

\textsuperscript{1} Dept of Quantitative Economics, Maastricht University, P.O.Box 616, 6200 MD Maastricht, The Netherlands.
\textsuperscript{2} e-mail: A.Koster@KE.Unimaas.NL; home page: http://www.unimaas.nl/~akoster/
a way that unacceptable interference is minimized is called the frequency assignment problem (FAP).

Research on the FAP already started in the 1960s. Hale [7] classified these approaches and modeled the FAP as mathematical optimization problems in two ways: the minimum order frequency assignment problem (MO-FAP), and the minimum span frequency assignment problem (MS-FAP). In the MO-FAP, we have to construct an interference-free assignment (i.e., an assignment without unacceptable interference) in which a minimum number of different frequencies are used, whereas in the MS-FAP the objective is to find an assignment in which the difference between minimum and maximum assigned frequency is as small as possible. Hale also pointed out that both problems are generalizations of the graph coloring problem.

The rapidly increase of wireless applications in the last 20 years has led to frequency assignment problems with other objectives. In case no interference-free assignment can be found, two ways remain: search for an interference-free assignment that serves as many customers as possible, or allow for an assignment with some interference. The problem in which we search for an interference-free assignment that serves as many customers as possible is referred to as the minimum blocking frequency assignment problem (MB-FAP). In case we allow interference, we either can minimize the maximum interference or minimize the cumulative interference. Minimization of the maximum interference can be done with a threshold value which results in a sequence of feasibility problems similar to the original one. The problem in which the cumulative interference is minimized is referred to as the minimum interference frequency assignment problem (MI-FAP).

For all four types of FAPs, researchers have proposed a variety of exact and heuristic methods from the fields of operations research, graph theory, and artificial intelligence. For an overview of these approaches we refer to [1, 9]. In this paper we concentrate on the MI-FAP. Exact methods, as well as lower bounds, however, are rarely described in the literature. A first attempt to obtain lower bounds is described in [13], where a quadratic programming relaxation is used. In [10] an integer programming formulation for the problem is presented and studied from a polyhedral point of view. Recently, in [11] a dynamic programming algorithm has been developed based on tree decomposition [3]. In this paper we combine the results of [10] and [11] in a procedure that generates a non-decreasing sequence of lower bounds.

The sequel of this paper is organized as follows. In Section 2, we describe the MI-FAP in more detail. Next, Section 3 is dedicated to a general description of the lower bound procedure. The exact solution of a (smaller) MI-FAP forms a part of the lower bound procedure. Therefore, in Section 4 we describe two exact methods based on integer linear programming and dynamic programming, respectively. The paper is closed with computational results in Section 5.

2 Problem Description

Mathematically, the MI-FAP is defined by the quadruple \((G, D, p, q)\), where \(G = (V, E)\) is the constraint graph. Each vertex represents a transmitter-receiver pair to which we have to assign
a frequency. Each edge indicates that for certain combinations of frequencies assigned to the adjacent vertices cause unacceptable interference. The set $D = \{ D_v : v \in V \}$ is the collection of all domains. For each of the $n$ vertices $v \in V$ in the graph the domain $D_v$ contains the frequencies available for that vertex. The last two components of the MI-FAP are two penalty functions $p$ and $q$. For each pair of adjacent vertices and corresponding choice of frequencies, the function $p$ determines the interference penalty. The function $q$ denotes the level of preference for all domain elements. The functions $p$ and $q$ are called the edge-penalty function and the vertex-penalty function, respectively. The objective of the problem is to select from every domain $D_v$ exactly one element in such a way that the total sum of the edge- and vertex-penalties is minimized.

The MI-FAP is formulated as a binary linear programming problem using the following binary variables for all $v \in V$, $d_v \in D_v$:

$$y(v, d_v) = \begin{cases} 1 & \text{if } d_v \in D_v \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

and for all $\{v, w\} \in E$, $d_v \in D_v$, $d_w \in D_w$

$$z(v, d_v, w, d_w) = \begin{cases} 1 & \text{if } (d_v, d_w) \in D_v \times D_w \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

The binary linear programming formulation then reads

$$\min \quad \sum_{\{v, w\} \in E} \sum_{d_v \in D_v} \sum_{d_w \in D_w} p(v, d_v, w, d_w) z(v, d_v, w, d_w) + \sum_{v \in V} \sum_{d_v \in D_v} q(v, d_v) y(v, d_v)$$

(1)

s.t. \quad \sum_{d_v \in D_v} y(v, d_v) = 1 \quad \forall v \in V 

(2)

$$\sum_{d_w \in D_w} z(v, d_v, w, d_w) = y(v, d_v) \quad \forall \{v, w\} \in E, d_v \in D_v$$

(3)

$$z(v, d_v, w, d_w) \in \{0, 1\} \quad \forall \{v, w\} \in E, d_v \in D_v, d_w \in D_w$$

(4)

$$y(v, d_v) \in \{0, 1\} \quad \forall v \in V, d_v \in D_v$$

(5)

Constraints (2) restrict the selection of frequencies from each domain to one. Constraints (3) enforce that the combination of values selected for an edge should be consistent with the values selected for the vertices of that edge.

The NP-hardness of the MI-FAP with domain sizes at least 3 follows from a reduction of the graph 3-colorability problem (cf. [6]). In [10] it is proved with a reduction from Maximum Satisfiability that the MI-FAP is NP-hard, even if all domains have size 2. The mathematical formulation of the MI-FAP is equivalent to the Partial Constraint Satisfaction Problem (PCSP).
3 Lower Bound Procedure

In [10] the polyhedral structure of the polytope defined by (2)-(5) is studied. The computational results show that only for instances with very small domains (at most 5 or 6 elements), the cutting plane algorithm is able to solve the problem. For real-life instances with 40 or more elements / frequencies the algorithm is not strong enough to even generate non-trivial lower bounds.

In [11] the graph structure of the MI-FAP is exploited more directly. For many combinatorial optimization problems based on graphs, the problem can be solved in polynomial time as long as the treewidth of the graph is limited by a constant (see [3] for an overview). In those cases a dynamic programming algorithm based on a tree decomposition of the graph solves the problem. With additional (pre)processing techniques, some of the instances can be solved in this way. However, the size of the domains causes that the algorithm is very time and memory consuming, which explains that the larger (more difficult) instances cannot be solved in this way.

As an alternative to the optimal solution, in [11] a lower bound procedure is developed in which the dynamic programming algorithm is used as a subroutine to solve MI-FAPs with far less frequencies as the original problem. This procedure can also be used in combination with the integer linear programming techniques of [10].

The idea behind the lower bound procedure is to combine subsets of frequencies to single frequencies. These subsets can be handled as frequencies of a new MI-FAP which has fewer frequencies than the original problem. As a consequence, the new MI-FAP can hopefully be solved to optimality with either integer linear programming or dynamic programming. By taking the minimum of the individual penalties of the frequencies in a subset as the vertex- and edge-penalties of the new MI-FAP, the optimal solution to the new MI-FAP is a lower bound for the original problem. We can extend this idea to an iterative method which provides a sequence of lower bounds for the original problem. An exact method has to be used as a subroutine to solve the MI-FAPs with the domain elements corresponding to subsets of frequencies.

The lower bounds derived with this procedure take advantage of the structure of the edge-penalties. For example, consider the matrix of edge-penalities given in Figure 1(a). The level of interference on this edge is 10 if the difference between the frequencies is less than 2. If we divide the frequencies in two groups \{1, 2\}, and \{3, 4\}, we obtain 4 blocks in the table of edge-penalities with (almost) the same values. In most cases there is no difference between the penalties as long as the pairs of frequencies are in the same block. Therefore, let us construct a new MI-FAP in which we have to assign either the subset \{1, 2\} or the subset \{3, 4\} to the vertices. The edge-penalities in this new MI-FAP are given by the minimum of the values in each block (see Figure 1(b)). Solving this substantially smaller problem provides a lower bound for the optimal value of the original problem. The quality of the lower bound depends on the size of the blocks: many small blocks will provide a better lower bound than a small number of large blocks. In most real-life instances the block structure of the penalty matrices arises naturally, since the available frequencies for a transmitter-receiver pair can be divided in groups of frequencies that are in the same part of the spectrum.
The above described idea can be formalized in the following procedure. We start with the original problem \( P = (G, D, p, q) \) and we partition for every vertex \( v \in V \) the domain \( D_v \) in an initial number of \( n_v \) subsets \( D^1_v, \ldots, D^{n_v}_v \). This partition is, for example, based on a natural partition of the frequencies in groups of frequencies that are in the same part of the radio spectrum.

Next, we construct a new MI-FAP \( P' = (G, D', p', q') \), with domains \( D'_v = \{1, \ldots, n_v\} \) for all vertices \( v \in V \), vertex-penalities

\[
q'(v,i) = \min_{d_v \in D'_v} q(v, d_v)
\]

for every vertex \( v \in V \), \( i \in D'_v \), and edge-penalities

\[
p'(v,i,w,j) = \min_{d_v \in D'_v} \min_{d_w \in D'_w} p(v, d_v, w, d_w)
\]

for all \( \{v, w\} \in E \), \( i \in D'_v \), \( j \in D'_w \). So, \( P' \) is defined on the same graph as \( P \), and the domains of \( P' \) correspond with the subsets \( D'_v \). Since the vertex and edge-penalities in \( P' \) are the minimum of the penalties in the corresponding subset(s), the optimal value of the problem \( P' \) provides a lower bound for the optimal value of the original problem \( P \).

The way to obtain a sequence of non-decreasing lower bounds is based on an iterative refinement of the domain-subsets. A partition \( D^1_v, \ldots, D^{n_v}_v \) of a domain \( D_v \) is called a refinement of another partition \( D^1_v, \ldots, D^{m}_v \), if for every subset \( D^i_v \), \( i = 1, \ldots, m \), there exists a subset \( D^j_v \), \( j \in \{1, \ldots, n\} \) in the second partition for which \( D^i_v \subseteq D^j_v \). If \( \hat{P} \) and \( \hat{P} \) are MI-FAPs corresponding to these partitions, then the value of the optimal solution of \( \hat{P} \) will be at least as high as the value of the optimal solution of \( \hat{P} \), which implies that \( \hat{P} \) provides a lower bound that is greater than or equal to the lower bound provided by \( \hat{P} \).

We can extend the procedure to obtain a lower bound to an algorithm that provides a sequence of non-decreasing lower bounds as follows. We construct a problem \( P^i \) which provides us with the first lower bound. Next, we refine the partition of the subsets, and again construct a MI-FAP \( P^i \) which hopefully provides us with a better lower bound. We can repeat the refinement of the
partition as long as the efforts to solve the problem \( P' \) is reasonable in both time and memory. A flowchart of this algorithm is presented in Figure 2.

![Flowchart](image)

Figure 2: Lower bound procedure

Whatever refinement procedure (i.e., for which vertices do we refine the partition, and how do we refine the partition) we apply, it is difficult to guarantee that the new lower bound will be strictly greater than the old lower bound. However, if for all vertices \( v \in V \), the domain-subset that corresponds to the optimal solution of \( P' \) is not partitioned in the refinement procedure, then the ‘old’ optimal solution is still optimal in the new problem \( P' \). This implies that a refinement can only be effective if at least one selected domain-subset is refined. Therefore, for each refinement we select one vertex \( v \), for which we partition the assigned subset. To speed up the process in practice, we do not apply the exact solution method after each single refinement, but after the refinement of the domains for a subset of the vertices \( S \subseteq V \).

For a partition of the assigned subset for a vertex \( v \in V \) we can compute an upper bound on the increase of the value of \( P' \). This upper bound is used as criteria to select a partition. Consider

<table>
<thead>
<tr>
<th>penalties</th>
<th>( D'_a )</th>
<th>( D'_o )</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( D' )</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>minimum</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 3: Example to illustrate the partition of assigned subsets
the example of Figure 3. Let \( D'_u = \{d^1_v, d^2_v\} \) be the assigned subset to \( v \), and let \( D'_u \) and \( D'_w \) be the assigned subsets to the neighbors \( u \) and \( w \), respectively. The total penalty incurred by this assignment is 0. However, if we either assign \( d^1_v \) or \( d^2_v \) to \( v \), then the total penalty will be one. Hence, partition of the subset may lead to an increase of the value of \( P' \). It cannot be guaranteed, however, since the new optimal assignment may select a subset other than \( \{d^1_v\} \) or \( \{d^2_v\} \).

In general, an upper bound on the increase of the optimal value by a partition of the assigned subset can be computed as follows. We restrict ourselves to a partition of the assigned domain-subset in two domain-subsets, but the procedure can easily be extended to a partition in more than two domain-subsets. The procedure can also be generalized to subsets of vertices instead of single vertices. Let \( v \in V' \) and \( D'_v \) be the domain-subset that corresponds to the optimal assignment. If we partition \( D'_v \) in \( A_v \) and \( D'_v \setminus A_v \), then the value of the problem \( P' \) will increase with at most \( \Delta \pi(v, A_v) \), which is defined as

\[
\Delta \pi(v, A_v) = \min\{\pi(v, A_v), \pi(v, D'_v \setminus A_v)\} - \pi(v, D'_v)
\]

where

\[
\pi(v, D) = \min_{d_v \in D} q(v, d_v) + \sum_{w \in N(v)} \min_{d_w \in D} \min_{d_u \in D_u} p(v, d_v, w, d_w)
\]

Among all partitions \( A_v, D'_v \setminus A_v \), the best partition, according to the value \( \Delta \pi(v, A_v) \), is \( A^*_v = \arg \max_{A_v \subset D'_v} \Delta \pi(v, A_v) \). If \( \Delta \pi(v, A^*_v) = 0 \) then no single refinement of the partition for vertex \( v \) will result in an increase of the lower bound for \( P \). Therefore, the subset \( S \) for which we will partition the assigned subset is given by the vertices for which \( \Delta \pi(v, A^*_v) > 0 \).

4 Exact Methods for the Subproblems

One of the two main subroutines of the lower bound procedure in the previous section is the use of an exact method to solve the smaller MIFAPs \( P' \). In this section we describe two exact methods that can be used for this purpose: integer linear programming and dynamic programming based on a tree decomposition of the constraint graph. We briefly describe the main characteristics of the algorithms, and the (dis)advantages in combination with the lower bound procedure.

4.1 Dynamic Programming - Tree Decomposition

One way to solve the subproblem is by the dynamic programming algorithm based on a tree decomposition of the constraint graph [11]. A tree decomposition of a graph is a tree consisting of induced subgraphs:
Definition 4.1 (Robertson and Seymour [12]) Let $G = (V, E)$ be a graph. A tree-decomposition is a pair $(T, \mathcal{X})$, where $T = (I, F)$ is a tree with nodes $I$ and edges $F$, and $\mathcal{X} = \{X_i : i \in I\}$ is a family of subsets of $V$, one for each node of $T$, such that

(i). for every edge $\{v, w\} \in E$, there is an $i \in I$ with $v \in X_i$ and $w \in X_i$, and

(ii). for all $i, j, k \in I$, if $j$ is on the path from $i$ to $k$ in $T$, then $X_i \cap X_k \subseteq X_j$.

The width of a tree decomposition is $\max_{i \in I} |X_i| - 1$. The treewidth of a graph $G$, denoted by $\text{tw}(G)$, is the minimum width over all possible tree decompositions of $G$.

The complexity of the construction of a tree decomposition of minimal treewidth is $\mathcal{NP}$-hard in general [2]. For fixed $k$, the question whether there exists a tree decomposition with width less than or equal to $k$ can be answered in polynomial time [4]. However, this algorithm is exponential in $k$, and is therefore impractical for graphs with larger treewidth.

A tree decomposition $(T, \mathcal{X})$ is the input of a dynamic programming algorithm that solves the MI-FAP in $\mathcal{O}(nd^{2k})$, where $n = |V|$, $d = \max_{v \in V} |D_v|$, and $k$ the width of the used tree decomposition. Starting with the leaves of the tree $T$, the algorithm computes for every node $i \in I$ all non-redundant partial assignments for the subgraph induced by the subtree rooted at the current node $i \in I$. Among all these partial assignments we can optimize over those that only differ for vertices not in $X_i$, since the assignment to those vertices does not influence the assignment of the remaining graph anymore. To compute all these non-redundant assignments, we can use the collections of non-redundant assignments for the children of the node $i$. For further details we refer to Koster, van Hoesel and Kolen [11].

4.2 Integer Linear Programming

Another way to solve the subproblem is by solving the integer linear program (1)-(5). In [10], the polyhedral structure of (1)-(5) is studied. Besides two lifting theorems, two specific classes of valid inequalities were derived, the cycle-inequalities and the clique-cycle inequalities. In [9] other classes of inequalities are derived based on the relationship with the boolean quadric polytope. All these inequalities can be used in a cutting plane algorithm (Cut and Branch) or Branch and Cut algorithm to solve the MI-FAP. Computational results in [9, 10] show that for small domains the cutting plane algorithm in which the cycle inequalities are added to the formulation performs very well.

5 Computational Results

In this section we compare the results of the stand-alone tree decomposition algorithm of [11] with the lower bound procedure with either dynamic programming or integer programming as a
subroutine. All implementations have been carried out in C++. The programs for the dynamic programming algorithm and the iterative version of the algorithm were running on a DEC 2100 A500MP workstation with 128Mb internal memory. We used the callable library of CPLEX 4.0 to solve (integer) linear programming problems. In Table 1 statistics (after preprocessing [11]) for the MI-FAP instances of the CALMA project [5] are presented. Three instances are already solved by preprocessing. The dynamic programming algorithm uses a heuristically determined tree decomposition. It solves 4 of the other instances. For the remaining instances the size of the domains is too large to solve them by dynamic programming. In Table 2 we report the results obtained with the lower bound procedure described in this paper. We start with either 2 or 4 subsets per domain. With the tree decomposition algorithm as subroutine, a lower bound within 12.5% of the best known value can be obtained for CELAR 07. For CELAR 06 the lower bound is only one below the optimal value, no matter whether we started with 2 or 4 subsets for every domain. The computation time, however, is for the start with 4 subsets substantially smaller than for 2 domain elements. Moreover, compared with the exact method, the computation time is reduced by 65%. For the instance CELAR 08 the results are less satisfactory with the tree decomposition algorithm as subroutine. For the instances GRAPH 11 and GRAPH 13 the width of the tree decomposition is simply too large to obtain any relevant results.

In case the polyhedral approach is used as a subroutine the results are more or less opposite. For the instances where the tree decomposition algorithm performed very well as subroutine, the results with integer programming are less satisfactory. The lower bounds obtained for CELAR 06 and CELAR 07 are not competitive with the previous results. For the other instances, however, the results for integer programming outperform the dynamic programming algorithm as subroutine. For both GRAPH 11 and GRAPH 13 lower bounds within 2% of the optimal values are obtained. Also for the instance CELAR 08, we can improve the lower bound substantially, from 33% to 57% of the best known solution.
<table>
<thead>
<tr>
<th>instance</th>
<th>after preprocessing</th>
<th>tree decomposition</th>
<th>best known</th>
</tr>
</thead>
<tbody>
<tr>
<td>CELAR 06</td>
<td>82 327 39.9</td>
<td>11 3389</td>
<td>27,102 3389</td>
</tr>
<tr>
<td>CELAR 07</td>
<td>162 764 34.6</td>
<td>17 4123</td>
<td>34,3592</td>
</tr>
<tr>
<td>CELAR 08</td>
<td>365 1539 39.4</td>
<td>18 262</td>
<td></td>
</tr>
<tr>
<td>CELAR 09</td>
<td>67 165 35.6 11391</td>
<td>7 15571</td>
<td>23 15571</td>
</tr>
<tr>
<td>CELAR 10</td>
<td>0 0 4556</td>
<td>solved by preprocessing</td>
<td>31,516</td>
</tr>
<tr>
<td>GRAPH 05</td>
<td>0 0 221</td>
<td>solved by preprocessing</td>
<td>221</td>
</tr>
<tr>
<td>GRAPH 06</td>
<td>119 348 16.2 4112</td>
<td>17 4123</td>
<td>29 4123</td>
</tr>
<tr>
<td>GRAPH 07</td>
<td>0 0 4324</td>
<td>solved by preprocessing</td>
<td>4324</td>
</tr>
<tr>
<td>GRAPH 11</td>
<td>340 1425 32.6 2553</td>
<td>104 3080</td>
<td>11 11827</td>
</tr>
<tr>
<td>GRAPH 12</td>
<td>61 123 15.3 11496</td>
<td>4 11827</td>
<td>11 11827</td>
</tr>
<tr>
<td>GRAPH 13</td>
<td>456 1874 38.1 8676</td>
<td>133 10110</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Statistics MI-FAP instances CALMA-project

<table>
<thead>
<tr>
<th>instance</th>
<th>initial # subsets</th>
<th>lower bounds</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>after preprocessing</td>
<td>tree decomp. CPU-time (sec)</td>
</tr>
<tr>
<td>CELAR 06</td>
<td>2 0 3388 13,734 2321 69,470 3389</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 0 3388 9,429 2146 67,904</td>
<td></td>
</tr>
<tr>
<td>CELAR 07</td>
<td>2 0 243066 259,022 180,325 256,418 34,3592</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 0 300000 275,736 -</td>
<td></td>
</tr>
<tr>
<td>CELAR 08</td>
<td>2 0 87 313,168 125 346,318 262</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 0 74 12,482 150 180,326</td>
<td></td>
</tr>
<tr>
<td>GRAPH 11</td>
<td>2 0 2553 - 2898 70,864 3080</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 0 - 3016 74,113</td>
<td></td>
</tr>
<tr>
<td>GRAPH 13</td>
<td>2 0 8676 - 9925 23,211 10110</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 0 - 9183 67,600</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Computational results lower bound procedure
References


