Frequency Assignment Problems: Benchmarks and Lower Bounds

Nick Dunkin, Stuart Allen, Derek Smith, Steve Hurley

May 21, 1998

Abstract

Benchmark problems for the Frequency Assignment Problem are uncommon in the literature. This scarcity is due to the highly sensitive nature of the information. Mobile communications vendors are understandably reluctant to give out information about the geographical location of transmitters in their networks. In the first part of this report a novel method for producing benchmark problem examples for the frequency assignment problem is introduced. This method uses probabilistic modelling to produce real looking transmitter locations on a fixed flat region. The output produced by the problem generator can then be used for any of a wide variety of solution techniques.

Lower bounding techniques can be used to assess the quality of assignments and to compare the relative merits of algorithms. In the second part of the report the main techniques used to obtain lower bounds for minimum span frequency assignment problems are presented. These include techniques based on Mathematical Programming. For cellular problems a dramatic simplification of these Mathematical Programming techniques is described. The method is shown to produce strong lower bounds. A potential application of the lower bounding technique to the assignment process itself is described.

Contents

1 Introduction

In an earlier report (Dunkin and Allen 1998) two aspects of the Frequency Assignment Problem (FAP) were studied. Methods of determining lower bounds for the span in minimum span frequency assignment problems were described and shown to give good results for a variety of examples. Such lower bounds allow the evaluation of assignments with respect to the problem representation, and also help in evaluating the relative merits of different algorithms. The report also contained a critical appraisal of the expressiveness of the usual binary constraint representation of the Frequency Assignment Problem. Alternatives to this binary constraint representation were proposed.

The current report continues this contribution to the evaluation of the methods and algorithms of frequency assignment. A Benchmark Generator is described which can be used to create sample data sets. These data sets contain information that is not usually available from network operators. The use of this Benchmark Generator will allow models and algorithms to be compared more easily, and allow comparisons between the work of different groups. The report also contains a more detailed account of lower bounding techniques than was contained in the previous report. In particular, in the case of cellular frequency assignment problems, a dramatic simplification of Mathematical Programming approaches is described. A possible use of this lower bounding technique in the actual assignment process is also outlined.

Part I
A Probabilistic Benchmark Generator for the Frequency Assignment Problem

2 Introduction

Benchmark examples for the Frequency Assignment Problem [9, 19] are rare due to the sensitive nature of the data. Network providers are understandably reluctant to give information relating to transmitter positions and powers. This can make research into efficient solution techniques for the frequency assignment problem difficult due to the limited availability of realistic problem instances. The benchmark problems that do exist have often been solved to the optimum solution and therefore are of reduced interest to researchers.

Useful example data for the frequency assignment problem would consist of a collection of geographical transmitter locations along with data pertaining to the transmitter such as power and direction. Obviously the most important of these is the geographical locations of the transmitters. The Benchmark Generator described in this document uses probability distributions over planar regions to construct realistic examples of the frequency assignment problem.

3 The Benchmark Generator

Throughout the frequency assignment literature there is a lack of benchmark problem examples of the frequency assignment problem. In this section we describe a probabilistic problem generator that is able to generate authentic looking problem instances of the frequency assignment problem.

3.1 A Suitable Representation

Research into finding solutions to the frequency assignment problem is wide ranging and utilises many different representation and solution techniques. As a result of this diversity the problem generator is not associated with any particular solution technique, to this end the output of the generator must be as generic as possible. In this way the generated benchmark problems may then be applied to any number of different representations and solution techniques, such as binary constraints or other higher order models. The most generic solution is for the output of the generator to consist of transmitter location coordinates over a fixed flat region of space.

3.2 Underlying Assumptions

We begin by assuming that the density of transmitter locations in a given region is directly proportional to the density of population or demand. You would expect to see many more transmitters per square mile in Central London than you would in the surrounding rural areas.

In order to model this density function over the fixed region R we construct Gaussians, centred at the location of *towns* in our fictional region. The height and width of the Gaussian

can be tailored to represent towns of different spread and population. As an example of this representation, a single large town centred at position $(25, 25)$ is shown in Figure 1.

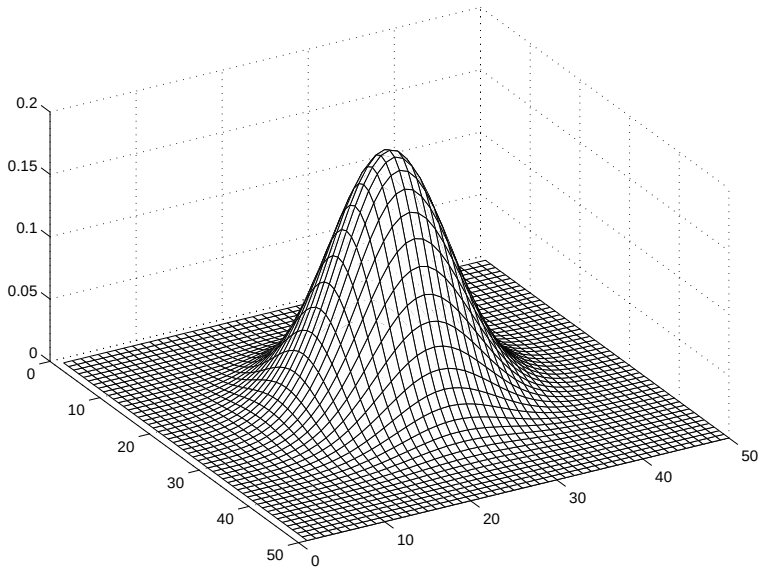

Figure 1: A density representation of a single large town

3.3 Building Up the Region

From this simple idea of representing a single town area, we can take a number of these Gaussians and build up a small fictional geographic region.

The Gaussian for each town can be constructed and then summed together to give a density function for the entire region R. In this way edges of towns that overlap give combined probability values, thus producing suitably more dense transmitter placings. A example of two towns overlapping is shown in Figure 2.

3.4 Generating Transmitter Locations

From the resulting density function and the addition of a small amount of random background noise we can generate the positions of transmitters placed to service the region R using the following probabilistic algorithm.

- For a given region R of fixed size, and
- for all points in the region (x, y).
- Place a transmitter at (x, y) with probability $P(x, y)$.

The probability $P(x, y)$ is the value of the density function at the point in the region (x, y).

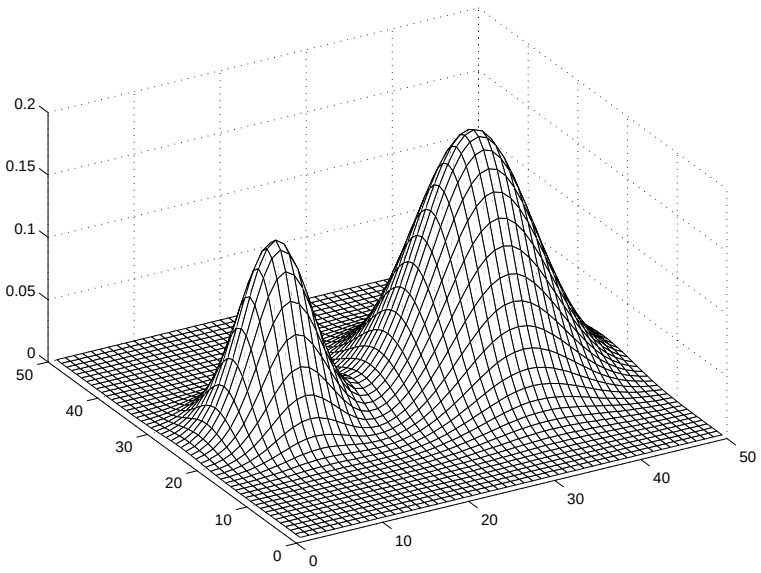

Figure 2: A density representation of two close towns of different sizes

3.5 An Example

In Figure 3 is shown the result of summing three Gaussians of various heights and widths at positions $(25, 10), (10, 40)$ and $(40, 50)$.

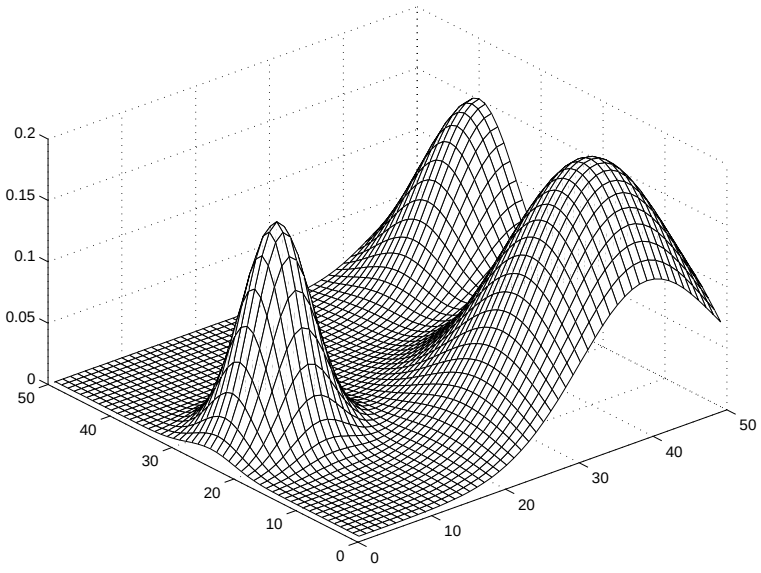

Figure 3: A density representation of three intersecting towns

This density/probability distribution can now be used to generate the transmitter locations shown in Figure 4.

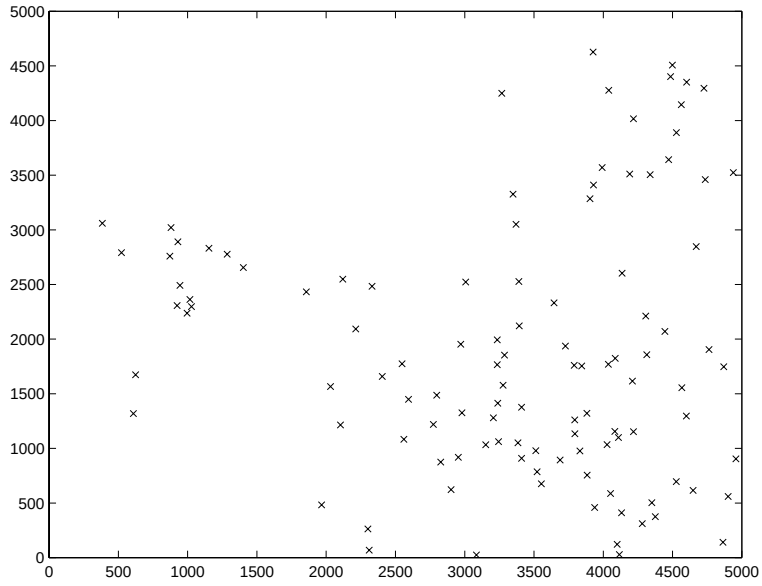

Figure 4: A graph of transmitters placements

4 Generating receiver positions

Some solution techniques used for the Frequency Assignment Problem require a collection of typical receiver points. These points can be used to check signal-to-noise ratios for any given assignment. Test receivers are best placed where there is expected to be the highest levels of interference in the region. The highest levels of interference are likely to be observed at points in the region that are equidistant between transmitters.

After the transmitters have been placed by the problem generator, a set of typical receiver points are generated for testing purposes. These points are displayed on the output as well as output to a special receiver file.

5 A receiver example

In Figure 5 the results of a two town plot are shown. The numbers represent the transmitter points while the small circles represent the chosen test receiver placements. Lines of equal distance are shown between the transmitters plotted on the region.

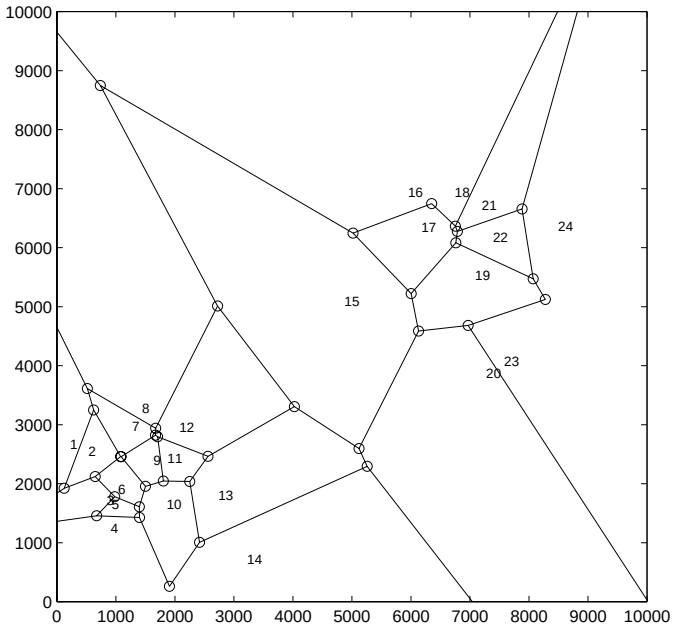

Figure 5: A graph of transmitters placements with test receivers

6 User Instructions

6.1 Introduction to the interface

In this section the user interface to the problem generator will be described.

6.1.1 Invoking the Problem Generator

The generator has been implemented in Matlab and therefore requires Matlab to be running on the user's machine. Having installed the benchmark generator's `.m` files in the Matlab source directory, the program can be initiated with

```
benchmark;
```

If the user has a parameter file from a previous run of the generator then the program can be initiated with.

```
benchmark('example1.prm');
```

This will invoke the generator with the parameters defined the file `example1.prm`. The format of the parameter file is described in Section 7.1. When using the parameter file option the user will still be able to control the number of problems generated.

6.1.2 Setting up the region

The user will then be asked to input the dimensions of the modelling region. During the perturbation part of the modelling process this region will be increased by 100 times so a suitable size would be,

```
Give dimensions of whole region

x-length of region : 100
y-length of region : 100
```

This will give a final region of approximately 10000 by 10000. Note that the placement of *towns* on this region will be with respect to the 100 by 100 region. The perturbation is a necessary part of the process, and introduces further randomness into the placement process.

6.1.3 Setting the background probability

Though most of the transmitters will be placed in the *town*-like areas of high probability there is a need to place some transmitters in the surrounding area with low probability, therefore a background probability value can be set to increase or decrease this likelihood. The background probability is enforced over the entire region.

The user is asked to input a *background* probability level with the following prompt,

```
Background probability : 0.001
```

The value of 0.001 gives good results but this value can be experimented with.

6.1.4 Setting the random seed

The transmitters are placed on the region with respect to the probability value at any point and the value generated by a random number generator. To ensure that successive passes of the generator produce different results this value should be changed each time the generator is used. Obviously to ensure that a particular problem instance can be regenerated the random seed should be recorded for future reference. The random seed is one of the parameters saved in the parameter file and is recalled when that parameter file is used. This ensures that when the generator is used with the parameter file input, problem instances can be regenerated.

The seed for the random number generator is input at the following prompt,

```
Random seed : 1234
```

6.1.5 Placing the towns

For any given problem instance the user can model a system with any number of *towns*. The number of *towns* required is input at the following prompt,

```
Number of towns : 1
```

For each town on the region, the user must input the centre of each town in respect to the original region size entered. In this example the original region size was `100` by `100`. The user must also input the *town*'s spread. The spread of a town determines how far out from the centre point the Gaussian will spread.

```
Number of towns : 1
x-center : 50
y-center : 50
x-length : 40
y-length : 40
height   : 0.02
cutoff   : 0.001
```

The `height` field sets the probability at the highest point in the Gaussian, the value of 0.02 seems to give good results and values any higher tend to produce far too dense a population of transmitters.

The final field, `cutoff`, controls how quickly the Gaussian falls off as it approaches the `x-length` and `y-length` points.

Figure 6 shows a Gaussian town centred at 50 x 50 with the following parameters,

```
x-length : 30
y-length : 30
height   : 0.02
cutoff   : 0.001
```

Notice that at 20 and 80 the Gaussian has a value of 0.001 - i.e. the cut off value specified in the definition.

By experimenting with these values, the user is able to create a very large number of different *town* types, from very wide, sparsely populated regions to very tight, densely populated regions.

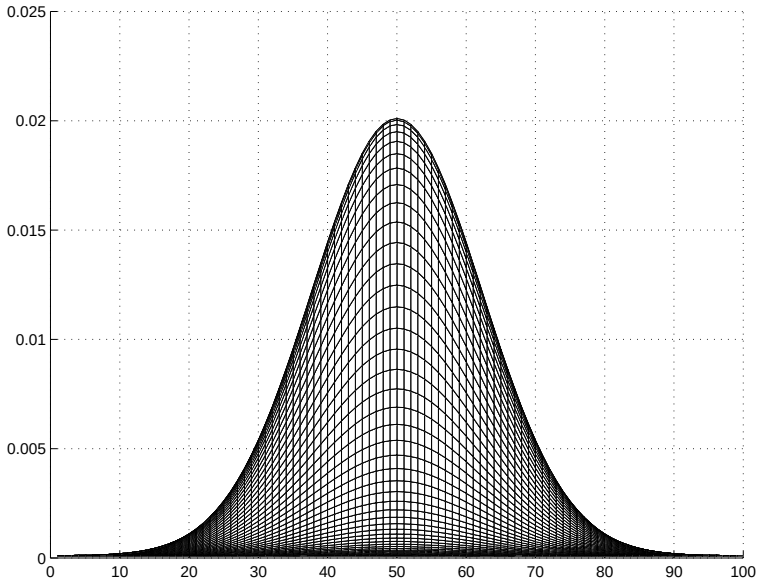

Figure 6: A Gaussian town with `height:0.02` and `cutoff:0.001`

6.1.6 Visualisation of the towns

After each town has been placed the generator will display a contoured image of the probability distribution over the region. This image is useful for checking that the *towns* are placed correctly and over-lapping where necessary. An example of this image for a single town placement is shown in Figure 7.

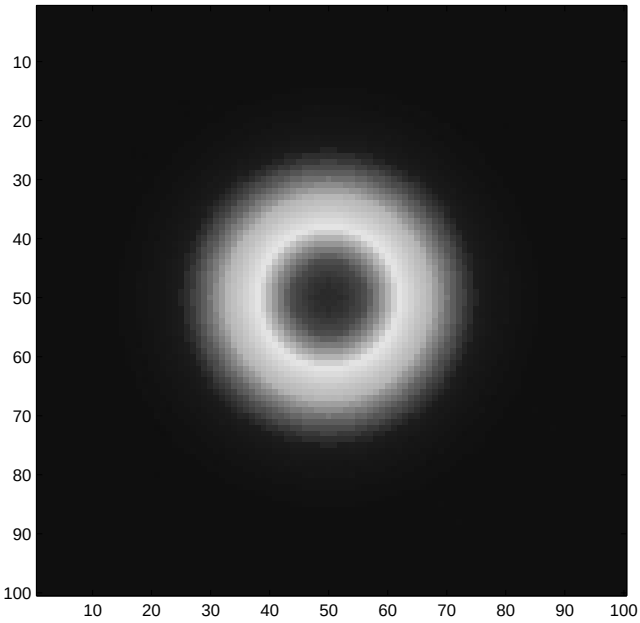

Figure 7: A probability density representation of a single large town

Assuming that everything is okay, the generator prompts the user to,

```
Press any key to place transmitters
```

6.1.7 Placing the transmitters

Any number of problems can be generated from a single probability distribution and the user is next asked to input how many problem instances are to be generated. The program prompts with,

```
Number of placement runs : 1
```

The generator outputs information for each problem instance to a file on disk, the user must give the root-name for these files,

```
Filename root            : example
```

The generator will now generate, for each problem instance, the following files

- `example#.prm` - The parameter file for problem number `#`
- `example#.trn` - The transmitter location file for problem number `#`

- `example#.rec` - The receiver location file for problem number `#`

As the generator creates each problem instance it displays important information about each instance. Remember that in addition to the transmitters placements the problem generator evaluates positions of likely high interference and at these points places test receivers positions. These receiver positions can be ignored but are useful in some solution methods.

For each placement the generator displays the number of transmitters placed and the number of test receiver points selected.

```
Placement 1.
44 transmitters placed.
72 receivers selected.
```

6.2 Graphical Output of the Problem

In order for the user to evaluate each problem instance for its suitability the program finally outputs the problem instance to the screen.

For each problem instance the generator displays the following diagram,

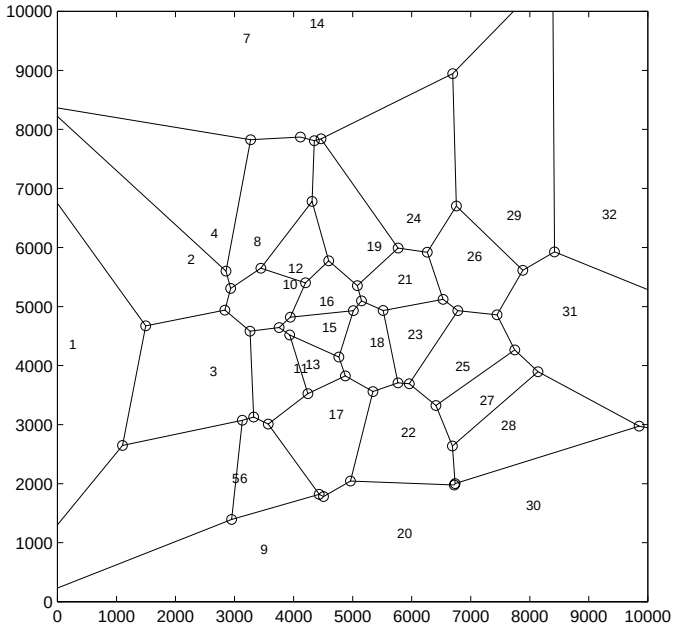

Figure 8: A placement of transmitters, resulting from a single large town

In Figure 8 the numbers are the positions of the transmitters, the small circles are the selected receiver positions and the lines divide the region up into possible cell areas. The division into cell areas assumes a propagation model that is not subject to noise and that every mobile receiver will always tune to its geographically closest transmitter.

7 The Benchmark Generator Output Files

The problem generator will, for each problem instance, create three output files of the following form. The file name route for these files is set by the user as described in the previous instructions.

7.1 The Parameter File

The first file created is the generator's parameter file. This file has a `.prm` extension. In the parameter file is recorded all the detail necessary to recreate one or more problem instances. To make problem instance regeneration a simpler task, the file name of the parameter file can be passed as an argument to the initial call to the benchmark program. For example `benchmark('example1.prm');` will invoke the generator and pass the arguments held in the `example1.prm` parameter file.

The files are all created automatically. Here is the structure of the `.prm` file.

```
% benchmark.m parameter file
% transmitter file : example1.trn
% receiver file    : example1.rec
x_reg 100
y_reg 100
background 0.0010
seed 1234
num_towns 1
50
50
40
40
0.0200
0.0001
```

7.2 The Transmitter Coordinates File

The next file created is the generator's transmitter coordinate file. This file has a `.trn` extension. After the sequence of comments the file contains the x and y coordinate of the transmitter location followed by an identifying number for the transmitter.

The files are all created automatically. Here is the structure of the `.trn` file.

```
% transmitter coordinates
% xmin xmax ymin ymax : 0    10000 0    10000
% parameter file   : example1.prm
% receiver file    : example1.rec
% format: x y trans_num

    198.97    4361.66        1
   2201.48    5804.10        2
   2582.21    3898.55        3
   2596.84    6234.71        4
   2959.73    2097.64        5
   3092.75    2083.20        6
   3143.85    9549.34        7
   3327.53    6098.47        8
   3437.25     884.33        9
   3816.77    5370.71       10
   3997.55    3958.49       11
   3909.55    5652.62       12
   4198.74    4020.81       13
   4273.67    9803.20       14
   4482.59    4650.80       15
   4436.24    5096.64       16
   4598.66    3175.62       17
   5289.67    4397.24       18
   5242.55    6026.44       19
   5757.04    1157.09       20
   5763.05    5459.71       21
   5819.32    2868.72       22
   5935.95    4530.47       23
   5912.34    6499.78       24
   6739.92    3994.20       25
   6941.86    5854.08       26
   7153.60    3407.90       27
   7517.58    2989.04       28
   7609.76    6546.25       29
   7940.51    1632.59       30
   8558.07    4916.24       31
   9223.23    6557.49       32
```

8 The Receiver Coordinates File

The final file created is the generator's receiver coordinate file. This file has a `.rec` extension. After the sequence of comments the file contains the x and y coordinate of the receiver location followed by an identifying number for the transmitter that the receiver is tuned to.

The files are all created automatically. Here is the structure of the `.rec` file.

```
% receiver coordinates
% xmin xmax ymin ymax : 0    10000 0    10000
% transmitter file : example1.trn
% parameter file   : example1.prm
% format: x y rec_num serving_trans_num

   1102.67    2648.38     1     1
   1495.95    4672.32     2     2
   2835.41    4939.95     3     3
   2856.46    5599.54     4     4
   2933.41    5305.20     5     8
   2950.57    1393.42     6     9
   3133.02    3073.99     7     3
   3262.21    4582.04     8    11
   3271.54    7825.82     9     7
   3323.80    3127.65    10     3
   3445.16    5649.22    11     8
   3570.68    3008.53    12     6
   3754.25    4645.02    13    15
   3934.21    4518.94    14    15
   3943.53    4820.09    15    10
   4114.45    7870.68    16     7
   4200.39    5400.68    17    16
   4242.31    3524.22    18    17
   4312.06    6780.86    19    12
   4350.79    7810.33    20    14
   4433.26    1819.47    21    17
   4463.57    7839.26    22    19
   4507.66    1781.76    23    17
   4594.43    5773.96    24    16
   4766.69    4143.86    25    18
   4876.90    3824.48    26    17
   4962.78    2042.94    27    20
   5014.11    4931.39    28    18
   5078.95    5353.80    29    16
   5149.38    5096.44    30    18
   5344.24    3560.13    31    22
   5516.09    4933.05    32    23
   5768.78    3707.23    33    22
```

5770.93	5989.33	34	21
5958.77	3693.90	35	25
6264.97	5918.42	36	24
6411.63	3323.48	37	25
6531.42	5121.97	38	26
6689.15	2636.73	39	27
6694.71	8946.02	40	14
6721.57	1978.94	41	22
6734.21	2000.63	42	28
6756.16	6701.59	43	26
6783.53	4930.37	44	25
7444.42	4858.61	45	26
7745.61	4264.70	46	31
7882.95	5614.33	47	26
8139.99	3897.48	48	27
8420.84	5927.27	49	32
9852.50	2972.90	50	28

Part II
Lower Bounds for Channel Assignment

9 Introduction

The study of lower bounds for channel assignment (Allen *et al.* to appear 1999, Gamst 1986, Lanfear 1989, Raychaudhuri 1985, Smith and Hurley 1997) is important for at least three reasons. Firstly, lower bounds can be used to show how good a particular assignment is, and whether it is capable of improvement. Secondly, they can be used to assess the effectiveness of particular algorithms. This is a much more powerful technique than simply comparing the results of algorithms one with another. Indeed, it is unfortunate that many papers on channel assignment algorithms have simply given results for a particular algorithm on sets of test data which may not be accessible to the reader. This can make it impossible for the reader to assess the effectiveness of the algorithm. Thirdly, it may be that identification of the particular structure in the problem that determines the best lower bound shows where the algorithm should concentrate its effort in order to find the best assignment.

This part of the report will concentrate on lower bounds for minimum span channel assignments. The study of lower bounds for the number of constraint violations in fixed spectrum problems with no feasible solution is much less well developed.

Express the channel assignment problem in its usual graph theoretic formulation. The transmitters are represented by the vertices $V(G)$ of a constraint graph G (Hurley *et al.* 1997):

Definition 1 *A* constraint graph *G is a finite, simple, undirected graph in which each edge v_iv_j ($v_i, v_j \in V(G)$) has an non-negative integer label ϕ_{ij}.*

Definition 2 *A* channel assignment *(or frequency assignment) in a constraint graph G is a mapping $f : V(G) \to F$ (where F is a set of consecutive integers $0, \ldots, K$) such that the constraints*

$$|f(v_i) - f(v_j)| > \phi_{ij}$$

are satisfied for all $v_iv_j \in E(G)$. Sometimes this is referred to as a zero-violation *assignment. If one or more of the inequalities are violated then f is an assignment with constraint violations. The elements of the set F can be referred to as* channels *(or as* frequencies*).*

Definition 3 *If K is a minimum over all zero-violation assignments then the assignment is a minimal assignment. This minimal value of K is the* minimum span *of G, denoted* $\mathrm{sp}(G)$.

Thus the span of an assignment is the difference between the largest channel used and the smallest channel used and $\mathrm{sp}(G)$ is the minimum span over all possible assignments. The object of this part of the report is to find a lower bound for $\mathrm{sp}(G)$, i.e. a value B such that $\mathrm{sp}(G) \geq B$. $B + 1$ represents a minimum number of consecutive channels that could possibly be used to assign G without constraint violations. Sometimes an assignment of span B can be found; sometimes no assignment of span B is possible. A lower bound B is only useful if B is close to $\mathrm{sp}(G)$.

10 Clique Bounds

The most common lower bound is based on cliques. The idea is borrowed from the theory of colourings of graphs. A *clique* of a graph G is a maximal complete subgraph of G. Thus every pair of vertices of the subgraph are adjacent, and the subgraph is not contained in any larger such subgraph (some authors omit this maximality requirement). For the purposes of channel assignment, a clique can be regarded as a set of transmitters for which there is a constraint between the frequencies assigned to any pair. Thus cliques tend to correspond to clusters of geographically close transmitters. As the edge labels of the constraint graph normally take several different values, it is possible to define several different levels of clique.

Definition 4 *A* level-p clique *of G is a complete subgraph in which every edge has label at least p, and which is not contained in any larger such complete subgraph.*

Thus a level-1 clique, for example, corresponds to a set of transmitters for which every pair of transmitters cannot be assigned the same channel, or a first adjacent channel.

Theorem 1 *If C_p is a level-p clique of a constraint graph G then*

$$\mathrm{sp}(G) \;\geq\; (p+1)(|V(C_p)|-1).$$

where $V(C_p)$ denotes the vertex set of the clique C_p.

Proof The minimum span of G cannot be less than the minimum span of the subgraph C_p of G. For any chosen minimal span assignment f of C_p renumber the vertices of C_p as $v_0, v_1, \ldots, v_{|V(C_p)|-1}$ in ascending order of the channel assigned to them (and arbitrary order for vertices assigned the same channel). The span of the assignment is the difference between the largest and the smallest channel used, i.e.

$$\begin{aligned}
\mathrm{sp}(C_p) &= f(v_{|V(C_p)|-1}) - f(v_0) \\
&= \sum_{j=0}^{|V(C_p)|-2} f(v_{j+1}) - f(v_j) \\
&\quad \text{(as all but two of the terms cancel in pairs)} \\
&\geq \sum_{j=0}^{|V(C_p)|-2} p+1 \\
&= (p+1)(|V(C_p)|-1).
\end{aligned}$$

□

Example 1 *The constraint graph shown in Fig. 9 has minimum span 11. A minimum span assignment is shown. Theorem 1 applied to the level-3 clique {2 3 5} gives* $\mathrm{sp}(G) \geq 8$. *Similarly, applying the theorem to the level-2 clique {2 3 4 5} gives* $\mathrm{sp}(G) \geq 9$. *It will be seen later that the clique bound is capable of improvement for this example.*

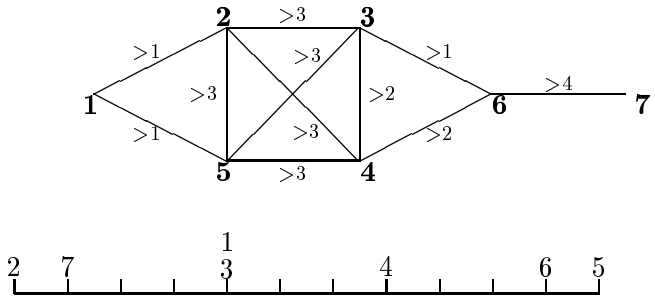

Figure 9: An Example Constraint Graph and Assignment

11 Travelling Salesman Bounds

A better bound can often be obtained using Hamiltonian paths. A Hamiltonian path in a graph G is a path through all of the vertices of the graph, visiting each vertex once and once only. Hamiltonian paths were first used by Raychaudhuri (1985) to derive bounds for the channel assignment problem. An account is also given by Smith and Hurley (1997). In order to calculate the bound it is first necessary to construct from G a weighted complete graph G' on the vertices of G. The weight c_{ij} of each edge v_iv_j of G' is given by:

$$\begin{array}{ll} c_{ij} = 0 & \text{if } v_iv_j \text{ is not an edge of } G, \\ c_{ij} = \phi_{ij} + 1 & \text{if edge } v_iv_j \text{ has label } \phi_{ij} \text{ in } G \ (\phi_{ij} = 0, 1, \ldots). \end{array}$$

Let $H(G')$ be the total weight of a minimum weight Hamiltonian path in G'.

Theorem 2 *If G is a constraint graph then*

$$\mathrm{sp}(G) \ \geq \ H(G')$$

Proof For any chosen minimal span assignment f of G number the vertices of G as $v_0, v_1, \ldots, v_{|V(G)|-1}$ in ascending order of the channel assigned to them (and arbitrary order for vertices assigned the same channel). Then $v_0, v_1, \ldots, v_{|V(G)|-1}$ is a Hamiltonian path H_f in G'. The span of the assignment is the difference between the largest and the smallest channel used, i.e.

$$\begin{array}{rcl} \mathrm{sp}(G) & = & f(v_{|V(G)|-1}) - f(v_0) \\ & = & \displaystyle\sum_{j=0}^{|V(G)|-2} f(v_{j+1}) - f(v_j) \\ & & \text{(as all but two of the terms cancel in pairs)} \\ & \geq & \displaystyle\sum_{j=0}^{|V(G)|-2} c_{v_{j+1}v_j} \\ & = & \text{the total weight of } H_f \\ & \geq & H(G'). \end{array}$$

□

The problem of determining the minimum weight of a Hamiltonian path in a graph is usually known as the open, symmetric travelling salesman problem (Volgenant and Jonker 1982). Thus the bound of Theorem 2 can be referred to as the *travelling salesman bound*.

It is important to realise that Theorem 2 should be applied to a subgraph of the constraint graph and not to the constraint graph itself. In most problems the application of the theorem to the full constraint graph gives a lower bound which is too small to be useful. Thus the theorem is applied with G taken to be the chosen subgraph of the full constraint graph. In order to obtain a strong bound, a suitable choice of subgraph is a clique or a clique with some vertices added. The question of how this subgraph should be obtained will be considered later.

A method due to Volgenant and Jonker (1982) and software described by Volgenant (1990)[1] can be used to calculate the travelling salesman bound. The software generally gives satisfactory results. Even when the algorithm does not converge in reasonable time, the difference between the lower and upper bound is small, so a good lower bound for $H(G')$ is usually obtained. However, the software is only applicable for subgraphs of up to 250 vertices, which does sometimes restrict its application.

Another bound, which is easier to calculate, is the *spanning tree bound*. A spanning tree in a graph is a connected subgraph of the graph which contains every vertex and no cycles. Let $S(G')$ denote the total weight of a minimum weight spanning tree in G'. As a Hamiltonian path is a spanning tree, it follows that $H(G') \geq S(G')$. Thus the following result is immediate:

Theorem 3 *If G is a constraint graph then*

$$\mathrm{sp}(G) \;\geq\; S(G')$$

Again the theorem should be applied to a suitable subgraph and not normally to the full constraint graph. The spanning tree bound is often not as strong as the travelling salesman bound, but is much easier to calculate. When applied to a clique it may be stronger than the clique bound. A simple greedy algorithm, known as Prim's Algorithm (Prim 1957), finds $S(G')$ immediately. An implementation is contained in FASoft (Hurley *et al.* 1997).

Returning to Example 1 it can be seen that applying both the spanning tree bound and the travelling salesman bound to the clique $\{2\ 3\ 4\ 5\}$ give lower bounds of 11, which equals $\mathrm{sp}(G)$.

12 An Example

The Philadelphia problems were first described by Anderson (1973) and have been studied by many authors, see for example (Gamst 1986), (Sivarajan *et al.* 1989). Optimal solutions to most of the standard variations have recently been found (Smith *et al.*, 1998), (Hurley *et al.* 1997). The problems are based on the area around Philadelphia, PA. The hexagonal geometry of 21 cells is shown in Fig. 10. A requirements vector $\mathbf{m}$ is used to describe the demand for frequencies in each cell. Thus m_i denotes the number of channels required by cell i and $\mathbf{m} = (8, 25, 8, 8, 8, 15, 18, 52, 77, 28, 13, 15, 31, 15, 36, 57, 28, 8, 10, 13, 8)$ in this example. Transmitters are considered to be located at cell centres and the distance between transmitters in adjacent cells is taken to be 1. Denote by d_k the smallest distance between transmitters

[1] The software is available on the World Wide Web. See:
http://www.mathematik.uni-kl.de/~ wwwwi/WWWWI/ORSEP/contents.html
ftp://www.mathematik.uni-kl.de/pub/Math/ORSEP/VOLGENAN.ZIP

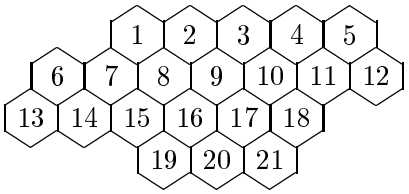

Figure 10: The Cellular Geometry of the Philadelphia Problems

which can use frequencies with a separation of k channels. In this example $d_0 = \sqrt{12}$, $d_1 = \sqrt{3}$, $d_2 = d_3 = d_4 = 1$ and $d_5 = 0$, so transmitters within the same cell (co-sited transmitters) must be separated by at least 5 channels.

The subgraph that must be used to obtain the lower bound is too large to apply the software described by Volgenant (1990). Thus an alternative method is necessary. The form of the travelling salesman bound used here was first obtained by Janssen and Kilakos (1996) using integer programming methods. They showed that if there are only two values of constraint between transmitters in different cells, then it is possible to classify the extreme points of a certain linear programming polytope. One of these extreme points corresponds to the bound described here.

The result is more easily described directly using Theorem 2. Let G_s denote the subgraph of the constraint graph corresponding to the transmitters in cell 9 and the six surrounding cells 2,3,8,10,16,17. Consider the graph G'_s which is a complete graph with no edges of weight 0. It has 275 vertices so $\mathrm{sp}(G_s)$ is at least 274. There are no edges of weight 1 incident with any vertex from cell 9. Thus the channels before and after any channel assigned to one of the 77 vertices in cell 9 must be unoccupied. These unassigned channels are distinct if the co-site value is 5. The minimum value of $H(G'_s)$ occurs when the first and last channels are assigned to vertices from cell 9. Then $H(G'_s) \geq 274 + 75 \times 2 + 2 \times 1 = 426$. Thus the minimal span for the complete problem is not less than 426. In fact an assignment of span 426 can be found using FASoft (Smith *et al.*, 1998), (Hurley *et al.* 1997). A frequency plan of the assignment can be found in (Smith *et al.*, 1998)[2]. Notice that the same bound can be derived (and the same assignment is valid) if the co-site value is 3 or 4 instead of 5.

13 Finding Cliques and Other Subgraphs

The following elementary result has already been used implicitly in the proof of Theorem 1.

Proposition 1 *If G'' is a subgraph of a constraint graph G then* $\mathrm{sp}(G) \geq \mathrm{sp}(G'')$.

Proof Suppose $\mathrm{sp}(G) < \mathrm{sp}(G'')$. Given a minimal span assignment of G, the channels assigned to the vertices of G'' form an assignment of G'' of span smaller than $\mathrm{sp}(G'')$, which would give a contradiction. □

It follows that if B is a lower bound for $\mathrm{sp}(G'')$, then B is also a lower bound for $\mathrm{sp}(G)$. The use of subgraphs is strictly necessary. For example, if Theorem 2 is applied to a full constraint

[2]The frequency plan can also be seen on the World Wide Web. See Example 3 at:
http://www.cs.cf.ac.uk/User/Steve.Hurley/plans.htm

graph, the number of edges of weight zero is often sufficiently large to allow $H(G')$ to equal zero. A good approach is to find a maximum level-p clique for several values of p, and then choose whichever value of p leads to the best lower bound. A better lower bound is sometimes obtained from a level-p clique with a number of additional vertices. A maximum level-p clique is simply a maximum clique in the subgraph of G obtained by removing all edges with label less than p. The remainder of this section considers how to find maximum cliques in a graph, how the procedure might be simplified for cellular problems and how vertices might be added to improve the bound.

13.1 Finding Maximum Cliques

A maximum clique in a graph is a complete subgraph (not contained in any larger complete subgraph), with the largest possible number of vertices. The problem of determining maximum cliques in a graph is NP-complete. This suggests that, as with the channel assignment problem itself, large problems cannot be solved exactly. However, for the graphs that arise in practice in channel assignment, it appears that the problems can be solved in reasonable time if the number of vertices is less than about 800. For larger graphs it would be necessary to resort to a heuristic procedure, such as that described by Gendreau *et al.* (1993). This would give a large clique which would not be guaranteed to be maximum, so the lower bound obtained might be weaker than that given by the actual maximum clique.

In order to find maximum cliques exactly for graphs with fewer than about 800 vertices, the partial enumeration algorithm presented by Carraghan and Pardalos (1990) gives good results. However, if it is to finish in reasonable time for the larger problems, a good ordering of the vertices should be used. The orderings introduced by Hale (1981) for sequential channel asssignment are effective in this application. Their effectiveness is discussed in more detail by Thiel *et al.* (1997).

13.2 Maximum Cliques in Cellular Problems

Determination of maximum cliques is much less computationally demanding when, as in the example in Section 12, there is a smaller number of transmitter locations and a demand vector which describes the number of transmitters at each site. As the co-site constraint is larger than all other constraints, it is easily seen that if a maximum clique includes one transmitter from a site, it will include all transmitters from the site. This fact can be used (Allen, unpublished) to dramatically reduce the computational requirement of the maximum clique algorithm.

13.3 Adding Vertices to Improve Lower Bounds

The difficulty of generating lower bounds for practical problems seems to vary considerably. For some practical problems the clique bound of Theorem 1 is attainable for an assignment and so is best possible. For other practical problems, the best lower bound is obtained from some level-p clique, but a travelling salesman or more sophisticated bound is necessary. When attempting to find the best possible lower bound, the hardest practical problems are those that require a subgraph larger than a clique. Such problems are not uncommon. It is possible to envisage at least three methods of adding vertices to a clique:

1. Take the chosen level-p clique as starting subgraph and repeatedly add vertices to the current subgraph as follows. For each vertex not in the subgraph, calculate the sum of the labels of the edges joining the vertex to the current subgraph. Add the vertex for which this sum is largest and re-evaluate the lower bound. If the lower bound does not decrease, continue.

2. Take the chosen level-p clique as starting subgraph and repeatedly add vertices to the current subgraph as follows. If the current subgraph can be assigned with a span equal to (or close to) the current bound, fix the assignment and attempt to extend the assignment to the full constraint graph. If any vertices are always involved in constraint violations with the subgraph, they should be added to the subgraph and the bound re-evaluated. If the lower bound does not decrease, continue.

3. Take the chosen level-p clique as starting subgraph and repeatedly add vertices to the current subgraph as follows. Assign the current subgraph, fix the assignment and calculate the available channels for each vertex not in the subgraph. Add the vertex or vertices with the smallest available number of channels. If the lower bound does not decrease, continue. This is the technique used in FASoft (Hurley *et al.* 1997).

Although all three methods have their merits and have proved successful on certain problems, each has its problems and none is guaranteed to work. They are also not easy to apply in practice. The development of these methods is a topic which merits further investigation.

14 Mathematical Programming

There are two reasons for using Mathematical Programming in lower bounding techniques for channel assignment. Firstly, it may be possible to more easily calculate bounds equivalent to or close to those described in previous sections. For example, the software described by Volgenant (1990) is restricted to a maximum of 250 vertices and does not always converge in a reasonable time. Sometimes the same result can be found using a simple linear program. Secondly, it is possible to find stronger bounds than the travelling salesman bound using these methods.

The approach of Janssen and Kilakos (1996) using Integer Programming has already been mentioned. Mathematical Programming techniques have also been used in the CALMA project[3]. The emphasis was on minimising the number of distinct frequencies used, rather than span minimization.

The travelling salesman problem can be formulated as an integer program. Let the graph G'_0 be formed from G' by the addition of a dummy vertex v_0 joined by an edge of weight 0 to each vertex of G'. This dummy vertex converts the open symmetric travelling salesman problem to a closed symmetric travelling salesman problem, where it is required to minimize the length of a circuit instead of the length of a path. If a minimal weight circuit is found in G'_0 then the vertex of weight 0 can be removed to give a minimum weight path in G'. Then $H(G')$ is equal to the solution of the following well known integer program for the closed symmetric travelling

[3]Information on the CALMA project can be found on the World Wide Web at http://www.win.tue.nl/math/bs/comb_opt/hurkens/calma.html

salesman problem (TSP):

$$\text{Minimize} \sum_{v_iv_j \in E(G_0')} c_{ij}x_{ij} \tag{1}$$

$$\text{subject to} \sum_{j:v_iv_j \in E(G_0')} x_{ij} = 2; \quad v_i \in V(G_0') \tag{2}$$

$$\sum_{v_i \in S, v_j \in V(G_0')\setminus S} x_{ij} \geq 2; \quad S \subset V(G_0') \tag{3}$$

$$x_{ij} \in \{0,1\}; \qquad v_iv_j \in E(G_0'). \tag{4}$$

The formulation gives a minimum weight Hamiltonian circuit in G_0' from which a minimal weight Hamiltonian path of weight $H(G')$ can be obtained by removing the vertex v_0. The integer variable x_{ij} is equal to 1 if edge v_iv_j is in the Hamiltonian circuit and equal to 0 otherwise. The total weight of the circuit to be minimised is given as 1. The equations 2 represent the requirement that there are two edges of the Hamiltonian circuit at each vertex. The inequalities 3 represent the so called subtour elimination constraints. These ensure that a single circuit is obtained, rather than the union of a number of disjoint circuits.

If this integer program is solved exactly the value of $H(G')$ is obtained. However, this may not be practical. One particular difficulty here is the memory required to store the subtour elimination inequalities, as there can be a very large number of them. One approach to making a solution practical is to relax the problem by weakening or removing one or more of the conditions. If the conditions are weakened it may be possible to find a solution of smaller total weight. Thus $H(G')$ may no longer be obtained, but instead a lower bound for $H(G')$ is found. This lower bound may be still be adequate for the purpose of deriving a strong lower bound for $\mathrm{sp}(G)$.

If the integrality constraint 4 is relaxed to

$$0 \leq x_{ij} \leq 1 \qquad v_iv_j \in E(G_0') \tag{5}$$

the linear (LP) relaxation of the integer program is obtained. This relaxation can be easy to solve and is generally a good lower bound. For random graphs the bound is, on average, within 1% of the exact value of $H(G')$. However, the memory requirement problem for the subtour elimination constraints remains.

An alternative is to relax the integer program by removing the subtour elimination constraints 3. This gives an integer program for the minimum weight perfect two-matching problem (PTMP) in G_0'. A perfect-two matching is a union of one or more circuits containing every vertex once and once only. An algorithm exists for finding a minimum weight perfect two-matching which is guaranteed to terminate in a time which is $O(|V(G)|^2|E(G)|)$ (Pekny and Miller 1994). A lower bound can also be obtained by replacing the integrality constraint 4 by the constraint 5 and using a standard Linear Programming package. For many subgraphs of constraint graphs consisting of a clique or a clique with some additional vertices, this lower bound is close to $H(G')$.

This LP relaxation of the minimum weight perfect two-matching problem, applied to a suitable subgraph, appears to be a simple, fast and robust method of obtaining good lower bounds for $\mathrm{sp}(G)$ in circumstances where the travelling salesman bound is strong. It is, of

course, never stronger than the travelling salesman bound. As an illustration of the effectiveness of the approach, consider the Philadelphia problem presented in Section 12. If we apply the LP relaxation of the minimum weight perfect two-matching problem to the same subgraph, generated by cells 2,3,8,9,10,16,17, then the tight lower bound of 426 is obtained. This approach has the advantage of being applicable to a wider range of problems than the rather specialised derivation presented in Section 12.

The linear programming approach described above is capable of improvement in two circumstances. The first circumstance is when the travelling salesman bound is inherently weak. This can arise due to the fact that the bound takes no account of constraints between non-consecutive vertices in the Hamiltonian path. The example of Section 12 shows that the travelling salesman bound is sometimes tight. It is certainly tight, for example, for a level-i clique where $\max_{\{i,j\}} c_{ij} \leq 2(i+1)$, as there exist no constraint violations between non-consecutive vertices of the Hamiltonian path. In this case the Hamiltonian path generates an assignment and the bound is best possible. However, there certainly exist many problems where the travelling salesman bound is not tight and a stronger bound is necessary. The second circumstance is where there is some uncertainty about the best subgraph to use for the bound. The travelling salesman bound (and most other bounds) have the property that their value reduces rapidly when more than a small number of critical vertices are added to the appropriate clique. This behaviour can be mitigated if the bound is improved. The bound will be improved here by adding additional constraints, referred to as frequency assignment constraints.

Associate a non-negative integer variable e_{ij} with each edge v_iv_j of the constraint graph G. The e_{ij} are chosen so that when an assignment is constructed from a Hamiltonian path $\{v_{i_1}, \ldots, v_{i_n}\}$ by setting

$$\begin{aligned} f(v_{i_1}) &= 0 \\ f(v_{i_j}) &= f(v_{i_{j-1}}) + c_{i_{j-1}i_j} + e_{i_{j-1}i_j} \qquad \text{for } j = 2, \ldots, n, \end{aligned}$$

the assignment will have no constraint violations. Then constraints between consecutive vertices on the Hamiltonian path no longer have to be met exactly, allowing constraints between non-consecutive vertices to be satisfied. In this case the value e_{ij} is referred to as the *excess* on the edge v_iv_j.

To formulate the frequency assignment constraints, make the following definitions. If P is a path $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$ with edge set $E(P)$, then let $X_P = x_{i_1i_2} + \cdots + x_{i_{k-1}i_k}$ and $E_P = e_{i_1i_2} + \cdots + e_{i_{k-1}i_k}$. Define the *deficit* of P as

$$d(P) = c_{i_1i_k} - (c_{i_1i_2} + \cdots + c_{i_{k-1}i_k}).$$

Let $\mathcal{P}(G')$ be the set of paths P of G' with $d(P) > 0$. Then, if $P \in \mathcal{P}(G')$ it is required that

$$X_P - (|E(P)| - 1) \leq \frac{E_P}{d(P)}.$$

If $X_P \leq (|E(P)| - 1)$ then E_P is unconstrained. If $X_P = |E(P)|$ (that is, if all edges of P are included in the Hamiltonian path), then the total excess on P must be at least as large as the deficit of P to ensure that the constraint between the end vertices of P is satisfied. This gives

the following integer programming formulation of the FAP:

$$\text{Minimize} \sum_{v_iv_j \in E(G_0')} c_{ij}x_{ij} + \sum_{v_iv_j \in E(G')} e_{ij} \tag{6}$$

$$\text{subject to} \sum_{j:v_iv_j \in E(G_0')} x_{ij} = 2; \quad v_i \in V(G_0') \tag{7}$$

$$\sum_{v_i \in S, v_j \in V(G_0')\setminus S} x_{ij} \geq 2; \quad S \subset V(G_0') \tag{8}$$

$$d(P)X_P - d(P)(|E(P)| - 1) - E_P \leq 0; \quad P \in \mathcal{P}(G') \tag{9}$$

$$x_{ij} \in \{0, 1\}; \qquad v_iv_j \in E(G_0') \tag{10}$$

$$e_{ij} \in \{0, 1, \ldots, c_{max}\}; \qquad v_iv_j \in E(G') \tag{11}$$

where $c_{max} = \max_{\{i,j\}} c_{ij}$. Note that when minimizing 6, e_{ij} will equal zero if $x_{ij} = 0$.

As with the formulation of the travelling salesman problem, a dummy vertex is used to express the problem in terms of a circuit rather than a path. The objective function in 6 is the total actual length of the path to be minimised. Equations 7 ensure that there are two edges of the circuit at each vertex. Inequalities 8 are the subtour elimination constraints. Inequalities 9 ensure that there are no constraint violations between non-consecutive vertices in the path. The fact that the variables x_{ij} are integers is expressed by 10 and the fact that the excesses are integers and at most equal to the maximum constraint value is expressed by 11

This integer program, if it can be solved, gives an exact solution of the channel assignment problem for the full constraint graph G. Although this is rarely possible, lower bounds can be derived as follows:

The integrality constraints 10, 11 are replaced by

$$0 \leq x_{ij} \leq 1; \qquad v_iv_j \in E(G_0') \tag{12}$$

$$0 \leq e_{ij} \leq c_{max}; \qquad v_iv_j \in E(G') \tag{13}$$

and a linear programming relaxation of the channel assignment problem is obtained. By also omitting the subtour elimination constraints 8, a linear programming relaxation of the formulation of the perfect two matching problem (PTMP), with extra frequency assignment constraints (PTMP+FAP) is obtained. This gives solutions in an acceptable time and appears to give an excellent bound when applied to a suitable subgraph, if it is practicable. Sometimes there are too many constraints in equation 9 and a subset of them must be chosen. This can be done by, for example, restricting the length of the paths $P \in \mathcal{P}(G')$ to 3 or 2.

The choice of which FAP constraints 9 to keep and which to eliminate may be critical in obtaining a strong bound. However, it appears that elimination of the subtour elimination constraints and relaxation of the integrality constraints makes little or no difference to the strength of the lower bound, for real frequency assignment problems. A level-0 clique is often a good candidate subgraph for this method, rather than a higher level clique. This is because the constraints in equation 9 tend to prevent constraint violations between non-consecutive vertices in the Hamiltonian path, which often occur if a travelling salesman bound is used with this level of clique.

Some results comparing this lower bound with previous methods and illustrating the improvement are given by Allen *et al.* (1997). These results are given for non-cellular problems. In the next section a simplification of this bound for cellular problems will be shown to be capable of generating a tight bound by solving a simple linear program.

15 An Example Using Mathematical Programming

The Mathematical Programming approach (PTMP+FAP) in the previous section has the potential to find the strongest possible bounds. However, the number of FAP constraints can, for some problems, be too large for a Linear Programming package to handle. In this section a simplification of the method will be described which, for cellular problems, dramatically reduces both the number of variables and the number of constraints. It does this by replacing the variables by certain sums of variables. The method will be illustated with a recent variation of the Philadelphia problem.

Tcha *et al.* (1997) proposed a new lower bound for the span of an assignment, extending the work of Gamst (1986). They show their bound to be tighter than the bound presented by Gamst and claim wider and easier real-world applicability. The variation of the Philadelphia problem presented is identical to that in Section 12 except that cells with centres at distance $\sqrt{3}$ require two channels separation. Tcha *et al.* show that their method gives a bound of 460 channels (i.e. $\mathrm{sp}(G) \geq 459$). This same bound of 459 can be obtained using the Linear Programming relaxation of the perfect two matching formulation, applied to the level-0 clique {1,2,3,7,8,9,10,15,16,17,19,20} with 360 vertices. Notice that the level-0 clique bound of Theorem 1 is only 359. The clique is too large to apply the software described by Volgenant (1990) to calculate the travelling salesman bound. Here it will be shown that a Linear Programming relaxation of PTMP+FAP gives a bound $\mathrm{sp}(G) \geq 524$, a result which is best possible.

Consider a cellular problem with transmitters located at a number of sites and with the number of transmitters at each site described by a demand vector $\mathbf{m}$. Let $\mathcal{C}_r$ denote site r, let $\mathcal{S}_{rs}$ be defined by:

$$\mathcal{S}_{rs} = \sum_{i,j} x_{ij} \quad (v_i \in \mathcal{C}_r,\ v_j \in \mathcal{C}_s)$$

and let $\mathcal{E}_{rs}$ be defined by:

$$\mathcal{E}_{rs} = \sum_{i,j} e_{ij} \quad (v_i \in \mathcal{C}_r,\ v_j \in \mathcal{C}_s).$$

Thus $\mathcal{S}_{rs}$ denotes the sum of the number of edges joining vertices at sites $\mathcal{C}_r$ and $\mathcal{C}_s$, and $\mathcal{E}_{rs}$ is the sum of the excesses on these edges.

Let $c = c_{ij},\ (i, j \in \mathcal{C}_r)$ be a constant cosite value and suppose $c > 2c_{ij}$ for all $(i \in \mathcal{C}_r,\ j \in \mathcal{C}_s,\ r \neq s)$. Then if $\mathcal{S}_{rs} > \min\{m_r, m_s\}$ there are at least $\mathcal{S}_{rs} - \min\{m_r, m_s\}$ edge disjoint paths of length 2 with the central vertex at one site and the other two vertices at the other site. Any such path requires a non-zero excess on one of the edges. By counting these excesses an inequality of the form:

$$(c - 2c_{rs})(\mathcal{S}_{rs} - \min\{m_r, m_s\}) \leq \mathcal{E}_{rs} \tag{14}$$

is obtained. This is a simplified form of the FAP constraints for paths of length 2.

Again use a dummy vertex v_0, this time considered to be at a dummy site 0 with $m_0 = 1$, in order to express the problem in terms of a circuit. As before, v_0 is joined by an edge of weight 0 to each vertex of G'. The condition that there are two edges at each vertex is expressed as:

$$\mathcal{S}_{r0} + 2\mathcal{S}_{rr} + \sum_{s\neq r,\ s\neq 0} \mathcal{S}_{rs} = 2m_r \quad (r \neq 0,\ m_r = |V(\mathcal{C}_r)|) \tag{15}$$

$$\sum_{s\neq 0} \mathcal{S}_{0s} = 2 \tag{16}$$

and the objective function to be minimised is

$$\sum_{r\neq 0} (c\mathcal{S}_{rr} + \mathcal{E}_{rr}) + \sum_{r\neq s,\ r,s\neq 0} (c_{rs}\mathcal{S}_{rs} + \mathcal{E}_{rs}) \tag{17}$$

where $c_{rs} = c_{ij}$ $(i \in \mathcal{C}_r, j \in \mathcal{C}_s)$. The variables are all integers.

In the Tcha *et al.* variation of the Philadelphia problem the level-0 clique shown in Fig. 11 is used.

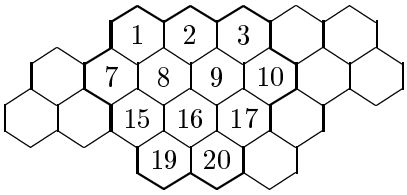

Figure 11: The cells forming a maximum level-0 clique

The objective function is

$$
\begin{array}{lllllllllllll}
5\mathcal{S}_{1\ 1} & + & 5\mathcal{S}_{2\ 2} & + & 5\mathcal{S}_{3\ 3} & + & 5\mathcal{S}_{7\ 7} & + & 5\mathcal{S}_{8\ 8} & + & 5\mathcal{S}_{9\ 9} & + \\
5\mathcal{S}_{10\ 10} & + & 5\mathcal{S}_{15\ 15} & + & 5\mathcal{S}_{16\ 16} & + & 5\mathcal{S}_{17\ 17} & + & 5\mathcal{S}_{19\ 19} & + & 5\mathcal{S}_{20\ 20} & + \\
2\mathcal{S}_{1\ 2} & + & \mathcal{S}_{1\ 3} & + & 2\mathcal{S}_{1\ 7} & + & 2\mathcal{S}_{1\ 8} & + & 2\mathcal{S}_{1\ 9} & + & \mathcal{S}_{1\ 10} & + \\
2\mathcal{S}_{1\ 15} & + & \mathcal{S}_{1\ 16} & + & \mathcal{S}_{1\ 17} & + & \mathcal{S}_{1\ 19} & + & \mathcal{S}_{1\ 20} & + & 2\mathcal{S}_{2\ 3} & + \\
2\mathcal{S}_{2\ 7} & + & 2\mathcal{S}_{2\ 8} & + & 2\mathcal{S}_{2\ 9} & + & 2\mathcal{S}_{2\ 10} & + & \mathcal{S}_{2\ 15} & + & 2\mathcal{S}_{2\ 16} & + \\
\mathcal{S}_{2\ 17} & + & \mathcal{S}_{2\ 19} & + & \mathcal{S}_{2\ 20} & + & \mathcal{S}_{3\ 7} & + & 2\mathcal{S}_{3\ 8} & + & 2\mathcal{S}_{3\ 9} & + \\
2\mathcal{S}_{3\ 10} & + & \mathcal{S}_{3\ 15} & + & \mathcal{S}_{3\ 16} & + & 2\mathcal{S}_{3\ 17} & + & \mathcal{S}_{3\ 19} & + & \mathcal{S}_{3\ 20} & + \\
2\mathcal{S}_{7\ 8} & + & \mathcal{S}_{7\ 9} & + & \mathcal{S}_{7\ 10} & + & 2\mathcal{S}_{7\ 15} & + & 2\mathcal{S}_{7\ 16} & + & \mathcal{S}_{7\ 17} & + \\
\mathcal{S}_{7\ 19} & + & \mathcal{S}_{7\ 20} & + & 2\mathcal{S}_{8\ 9} & + & \mathcal{S}_{8\ 10} & + & 2\mathcal{S}_{8\ 15} & + & 2\mathcal{S}_{8\ 16} & + \\
2\mathcal{S}_{8\ 17} & + & 2\mathcal{S}_{8\ 19} & + & \mathcal{S}_{8\ 20} & + & 2\mathcal{S}_{9\ 10} & + & 2\mathcal{S}_{9\ 15} & + & 2\mathcal{S}_{9\ 16} & + \\
2\mathcal{S}_{9\ 17} & + & \mathcal{S}_{9\ 19} & + & 2\mathcal{S}_{9\ 20} & + & \mathcal{S}_{10\ 15} & + & 2\mathcal{S}_{10\ 16} & + & 2\mathcal{S}_{10\ 17} & + \\
\mathcal{S}_{10\ 19} & + & \mathcal{S}_{10\ 20} & + & 2\mathcal{S}_{15\ 16} & + & \mathcal{S}_{15\ 17} & + & 2\mathcal{S}_{15\ 19} & + & 2\mathcal{S}_{15\ 20} & + \\
2\mathcal{S}_{16\ 17} & + & 2\mathcal{S}_{16\ 19} & + & 2\mathcal{S}_{16\ 20} & + & 2\mathcal{S}_{17\ 19} & + & 2\mathcal{S}_{17\ 20} & + & 2\mathcal{S}_{19\ 20} & + \\
\mathcal{E}_{1\ 1} & + & \mathcal{E}_{2\ 2} & + & \mathcal{E}_{3\ 3} & + & \mathcal{E}_{7\ 7} & + & \mathcal{E}_{8\ 8} & + & \mathcal{E}_{9\ 9} & + \\
\mathcal{E}_{10\ 10} & + & \mathcal{E}_{15\ 15} & + & \mathcal{E}_{16\ 16} & + & \mathcal{E}_{17\ 17} & + & \mathcal{S}_{19\ 19} & + & \mathcal{E}_{20\ 20} & + \\
\mathcal{E}_{1\ 2} & + & \mathcal{E}_{1\ 3} & + & \mathcal{E}_{1\ 7} & + & \mathcal{E}_{1\ 8} & + & \mathcal{E}_{1\ 9} & + & \mathcal{E}_{1\ 10} & + \\
\mathcal{E}_{1\ 15} & + & \mathcal{E}_{1\ 16} & + & \mathcal{E}_{1\ 17} & + & \mathcal{E}_{1\ 19} & + & \mathcal{E}_{1\ 20} & + & \mathcal{E}_{2\ 3} & + \\
\mathcal{E}_{2\ 7} & + & \mathcal{E}_{2\ 8} & + & \mathcal{E}_{2\ 9} & + & \mathcal{E}_{2\ 10} & + & \mathcal{E}_{2\ 15} & + & \mathcal{E}_{2\ 16} & + \\
\mathcal{E}_{2\ 17} & + & \mathcal{E}_{2\ 19} & + & \mathcal{E}_{2\ 20} & + & \mathcal{E}_{3\ 7} & + & \mathcal{E}_{3\ 8} & + & \mathcal{E}_{3\ 9} & + \\
\mathcal{E}_{3\ 10} & + & \mathcal{E}_{3\ 15} & + & \mathcal{E}_{3\ 16} & + & \mathcal{E}_{3\ 17} & + & \mathcal{E}_{3\ 19} & + & \mathcal{E}_{3\ 20} & + \\
\mathcal{E}_{7\ 8} & + & \mathcal{E}_{7\ 9} & + & \mathcal{E}_{7\ 10} & + & \mathcal{E}_{7\ 15} & + & \mathcal{E}_{7\ 16} & + & \mathcal{E}_{7\ 17} & + \\
\mathcal{E}_{7\ 19} & + & \mathcal{E}_{7\ 20} & + & \mathcal{E}_{8\ 9} & + & \mathcal{E}_{8\ 10} & + & \mathcal{E}_{8\ 15} & + & \mathcal{E}_{8\ 16} & + \\
\mathcal{E}_{8\ 17} & + & \mathcal{E}_{8\ 19} & + & \mathcal{E}_{8\ 20} & + & \mathcal{E}_{9\ 10} & + & \mathcal{E}_{9\ 15} & + & \mathcal{E}_{9\ 16} & + \\
\mathcal{E}_{9\ 17} & + & \mathcal{E}_{9\ 19} & + & \mathcal{E}_{9\ 20} & + & \mathcal{E}_{10\ 15} & + & \mathcal{E}_{10\ 16} & + & \mathcal{E}_{10\ 17} & + \\
\mathcal{E}_{10\ 19} & + & \mathcal{E}_{10\ 20} & + & \mathcal{E}_{15\ 16} & + & \mathcal{E}_{15\ 17} & + & \mathcal{E}_{15\ 19} & + & \mathcal{E}_{15\ 20} & + \\
\mathcal{E}_{16\ 17} & + & \mathcal{E}_{16\ 19} & + & \mathcal{E}_{16\ 20} & + & \mathcal{E}_{17\ 19} & + & \mathcal{E}_{17\ 20} & + & \mathcal{E}_{19\ 20}. &
\end{array}
$$

The conditions for two edges at each vertex, including the condition for two edges at the dummy vertex, are expressed as:

$$
\begin{aligned}
\mathcal{S}_{0\ 1} + 2\mathcal{S}_{1\ 1} + \mathcal{S}_{1\ 2} + \mathcal{S}_{1\ 3} + \mathcal{S}_{1\ 7} + \mathcal{S}_{1\ 8} + \mathcal{S}_{1\ 9} + \mathcal{S}_{1\ 10} + \\
\mathcal{S}_{1\ 15} + \mathcal{S}_{1\ 16} + \mathcal{S}_{1\ 17} + \mathcal{S}_{1\ 19} + \mathcal{S}_{1\ 20} &= 16 \\
\mathcal{S}_{0\ 2} + \mathcal{S}_{1\ 2} + 2\mathcal{S}_{2\ 2} + \mathcal{S}_{2\ 3} + \mathcal{S}_{2\ 7} + \mathcal{S}_{2\ 8} + \mathcal{S}_{2\ 9} + \mathcal{S}_{2\ 10} + \\
\mathcal{S}_{2\ 15} + \mathcal{S}_{2\ 16} + \mathcal{S}_{2\ 17} + \mathcal{S}_{2\ 19} + \mathcal{S}_{2\ 20} &= 50 \\
\mathcal{S}_{0\ 3} + \mathcal{S}_{1\ 3} + \mathcal{S}_{2\ 3} + 2\mathcal{S}_{3\ 3} + \mathcal{S}_{3\ 7} + \mathcal{S}_{3\ 8} + \mathcal{S}_{3\ 9} + \mathcal{S}_{3\ 10} + \\
\mathcal{S}_{3\ 15} + \mathcal{S}_{3\ 16} + \mathcal{S}_{3\ 17} + \mathcal{S}_{3\ 19} + \mathcal{S}_{3\ 20} &= 16 \\
\mathcal{S}_{0\ 7} + \mathcal{S}_{1\ 7} + \mathcal{S}_{2\ 7} + \mathcal{S}_{3\ 7} + 2\mathcal{S}_{7\ 7} + \mathcal{S}_{7\ 8} + \mathcal{S}_{7\ 9} + \mathcal{S}_{7\ 10} + \\
\mathcal{S}_{7\ 15} + \mathcal{S}_{7\ 16} + \mathcal{S}_{7\ 17} + \mathcal{S}_{7\ 19} + \mathcal{S}_{7\ 20} &= 36 \\
\mathcal{S}_{0\ 8} + \mathcal{S}_{1\ 8} + \mathcal{S}_{2\ 8} + \mathcal{S}_{3\ 8} + \mathcal{S}_{7\ 8} + 2\mathcal{S}_{8\ 8} + \mathcal{S}_{8\ 9} + \mathcal{S}_{8\ 10} + \\
\mathcal{S}_{8\ 15} + \mathcal{S}_{8\ 16} + \mathcal{S}_{8\ 17} + \mathcal{S}_{8\ 19} + \mathcal{S}_{8\ 20} &= 104 \\
\mathcal{S}_{0\ 9} + \mathcal{S}_{1\ 9} + \mathcal{S}_{2\ 9} + \mathcal{S}_{3\ 9} + \mathcal{S}_{7\ 9} + \mathcal{S}_{8\ 9} + 2\mathcal{S}_{9\ 9} + \mathcal{S}_{9\ 10} + \\
\mathcal{S}_{9\ 15} + \mathcal{S}_{9\ 16} + \mathcal{S}_{9\ 17} + \mathcal{S}_{9\ 19} + \mathcal{S}_{9\ 20} &= 154
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{S}_{0\ 10} + \mathcal{S}_{1\ 10} + \mathcal{S}_{2\ 10} + \mathcal{S}_{3\ 10} + \mathcal{S}_{7\ 10} + \mathcal{S}_{8\ 10} + \mathcal{S}_{9\ 10} + 2\mathcal{S}_{10\ 10} + \\
\mathcal{S}_{10\ 15} + \mathcal{S}_{10\ 16} + \mathcal{S}_{10\ 17} + \mathcal{S}_{10\ 19} + \mathcal{S}_{10\ 20} &= 56 \\
\mathcal{S}_{0\ 15} + \mathcal{S}_{1\ 15} + \mathcal{S}_{2\ 15} + \mathcal{S}_{3\ 15} + \mathcal{S}_{7\ 15} + \mathcal{S}_{8\ 15} + \mathcal{S}_{9\ 15} + \mathcal{S}_{10\ 15} + \\
2\mathcal{S}_{15\ 15} + \mathcal{S}_{15\ 16} + \mathcal{S}_{15\ 17} + \mathcal{S}_{15\ 19} + \mathcal{S}_{15\ 20} &= 72 \\
\mathcal{S}_{0\ 16} + \mathcal{S}_{1\ 16} + \mathcal{S}_{2\ 16} + \mathcal{S}_{3\ 16} + \mathcal{S}_{7\ 16} + \mathcal{S}_{8\ 16} + \mathcal{S}_{9\ 16} + \mathcal{S}_{10\ 16} + \\
\mathcal{S}_{15\ 16} + 2\mathcal{S}_{16\ 16} + \mathcal{S}_{16\ 17} + \mathcal{S}_{16\ 19} + \mathcal{S}_{16\ 20} &= 114 \\
\mathcal{S}_{0\ 17} + \mathcal{S}_{1\ 17} + \mathcal{S}_{2\ 17} + \mathcal{S}_{3\ 17} + \mathcal{S}_{7\ 17} + \mathcal{S}_{8\ 17} + \mathcal{S}_{9\ 17} + \mathcal{S}_{10\ 17} + \\
\mathcal{S}_{15\ 17} + \mathcal{S}_{16\ 17} + 2\mathcal{S}_{17\ 17} + \mathcal{S}_{17\ 19} + \mathcal{S}_{17\ 20} &= 56 \\
\mathcal{S}_{0\ 19} + \mathcal{S}_{1\ 19} + \mathcal{S}_{2\ 19} + \mathcal{S}_{3\ 19} + \mathcal{S}_{7\ 19} + \mathcal{S}_{8\ 19} + \mathcal{S}_{9\ 19} + \mathcal{S}_{10\ 19} + \\
\mathcal{S}_{15\ 19} + \mathcal{S}_{16\ 19} + \mathcal{S}_{17\ 19} + 2\mathcal{S}_{19\ 19} + \mathcal{S}_{19\ 20} &= 20 \\
\mathcal{S}_{0\ 20} + \mathcal{S}_{1\ 20} + \mathcal{S}_{2\ 20} + \mathcal{S}_{3\ 20} + \mathcal{S}_{7\ 20} + \mathcal{S}_{8\ 20} + \mathcal{S}_{9\ 20} + \mathcal{S}_{10\ 20} + \\
\mathcal{S}_{15\ 20} + \mathcal{S}_{16\ 20} + \mathcal{S}_{17\ 20} + \mathcal{S}_{19\ 20} + 2\mathcal{S}_{20\ 20} &= 26 \\
\mathcal{S}_{0\ 1} + \mathcal{S}_{0\ 2} + \mathcal{S}_{0\ 3} + \mathcal{S}_{0\ 7} + \mathcal{S}_{0\ 8} + \mathcal{S}_{0\ 9} + \mathcal{S}_{0\ 10} + \\
\mathcal{S}_{0\ 15} + \mathcal{S}_{0\ 16} + \mathcal{S}_{0\ 17} + \mathcal{S}_{0\ 19} + \mathcal{S}_{0\ 20} &= 2
\end{aligned}
$$

and a selection of FAP constraints is

$$
\begin{aligned}
\mathcal{S}_{8\ 9} - \mathcal{E}_{8\ 9} - 52 &\leq 0 \\
\mathcal{S}_{8\ 16} - \mathcal{E}_{8\ 16} - 52 &\leq 0 \\
\mathcal{S}_{9\ 16} - \mathcal{E}_{9\ 16} - 57 &\leq 0 \\
3\mathcal{S}_{7\ 9} - \mathcal{E}_{7\ 9} - 54 &\leq 0 \\
3\mathcal{S}_{9\ 19} - \mathcal{E}_{9\ 19} - 30 &\leq 0 \\
3\mathcal{S}_{8\ 10} - \mathcal{E}_{8\ 10} - 84 &\leq 0 \\
3\mathcal{S}_{8\ 20} - \mathcal{E}_{8\ 20} - 39 &\leq 0 \\
3\mathcal{S}_{1\ 16} - \mathcal{E}_{1\ 16} - 24 &\leq 0 \\
3\mathcal{S}_{3\ 16} - \mathcal{E}_{3\ 16} - 24 &\leq 0 \\
3\mathcal{S}_{2\ 15} - \mathcal{E}_{2\ 15} - 75 &\leq 0 \\
3\mathcal{S}_{2\ 17} - \mathcal{E}_{2\ 17} - 75 &\leq 0 \\
3\mathcal{S}_{15\ 17} - \mathcal{E}_{15\ 17} - 84 &\leq 0
\end{aligned}
$$

If this linear program is run on a standard Mathematical Programing package a lower bound of 524 is obtained. This bound is obtained whether it is run as integer program (with the condition that all variables be integers added) or as a linear program (without such a condition). In both cases the result is given in 0.24 seconds (using CPLEX on a 133MHz 64Mb Pentium PC). The same is true if the full set of FAP constraints is generated. In fact the lower bound of 524 is best possible, as FASoft (Smith *et al.*, 1998), (Hurley *et al.* 1997) is able to find an assignment of span 524.

The equations and inequalities appear extensive. However, they can be directly generated from the matrix of elements c_{rs} and the vector $\mathbf{m}$ in a form suitable for input to a Linear Programming package.

The method has proved succesful on other problems. It can, for example, be applied to the original Philadelphia problem described in Section 12. If it is applied to the level-0 clique generated by cells {1,2,3,7,8,9,10,15,16,17,19,20} then the lower bound of 426 is again found. This has the advantage that the subgraph is a standard clique which can be found directly using a maximum clique algorithm.

For cellular problems the method appears to be the best of those described in this report. The requirement for the method to give a strong bound is that there must exist a suitable subgraph, with a minimum span which determines the minimum span of G.

16 Assigning Cliques Using Integer Programming Bounds

The work of the previous section was directed towards the determination of a lower bound for the span of an assignment in a cellular problem. In the examples attempted so far, with the subgraph chosen to be a level-0 clique, an integer solution to the Mathematical Programming problem has been obtained.

This opens up an interesting possibility. As the solution to the integer programming formulation is obtained quickly, it may provide a fast and effective means of assigning the clique. Essentially, if all paths P with deficit $d(P) > 0$ have been included in the formulation, then the integer solution leads constructively to a perfect 2-matching in G_0', with no constraint violations between non-adjacent vertices in the matching. If the perfect 2-matching is a union of circuits rather than a single circuit, a simple heuristic is then used to merge circuits wherever possible. If a single circuit is obtained without creating new constraint violations, removal of the dummy vertex gives an assignment of minimum span. If a single circuit is not obtained, the circuits can still be merged, at the cost of an increase in span. However, good assignments may still be obtained quickly in this case as well.

The restriction of the method to level-0 (or higher level) cliques in cellular problems does of course limit its applicability. However, if the technique proves effective in practice, it may well be useful for problems which are best solved by assigning a subgraph first, fixing the assignment and then extending it to the whole problem. It has already been established (Smith *et al.* 1998) that some problems are best solved by this subgraph approach. The method can certainly be expected to be faster than the simulated annealing or tabu search heuristics used currently (Hurley *et al.* 1997). Whether the assignments are at least as good as those obtained by such heuristics remains to be evaluated.

The requirement that all paths P with deficit $d(P) > 0$ be included in the formulation is quite commonly satisfied in cellular frequency assignment problems. For example, if $max_{i \neq j} c_{ij} = 2$ and $c = 3$ then the only paths with potential deficit in a level-0 clique are those included in the formulation.

The details of the assignment procedure will now be given. Suppose that an integer solution of the formulation given in section 15 (including all paths P with deficit $d(P) > 0$) is available. There are four basic steps:

1. **Construct the perfect 2-matching**
   Given an integer solution to the linear program (or a solution to the integer program) described in section 15, there is an integer $\mathcal{S}_{rs}$ and an integer $\mathcal{E}_{rs}$ for each pair of sites $\mathcal{C}_r$ and $\mathcal{C}_s$, where $(r \neq s)$. Associated with each pair r,s is a pair of integers m_r and

m_s representing the number of transmitters at site $\mathcal{C}_r$ and site $\mathcal{C}_s$ respectively. There is also an integer $\mathcal{S}_{rr}$ for each site $\mathcal{C}_r$. Consider first the construction of the subgraph of the matching whose edges have one vertex from site $\mathcal{C}_r$ and one vertex from site $\mathcal{C}_s$. The subgraph has $\mathcal{S}_{rs}$ edges. Start with a set M of $\mathcal{S}_{rs}$ disjoint edges, with one end associated with the site $\mathcal{C}_r$ and one end associated with the site $\mathcal{C}_s$. Denote the vertices associated with site $\mathcal{C}_r$ by $u_1, u_2, u_3 \ldots$ and the vertices associated with site $\mathcal{C}_s$ by $w_1, w_2, w_3 \ldots$. If $\mathcal{S}_{rs} > m_r$ then merge $\mathcal{S}_{rs} - m_r$ pairs of vertices $(v_1, v_2), (v_3, v_4), (v_5, v_6), \ldots$. Similarly if $\mathcal{S}_{rs} > m_s$ then merge $\mathcal{S}_{rs} - m_s$ pairs of vertices $(w_2, w_3), (w_4, w_5), (w_6, w_7), \ldots$. The subgraph formed is then either a union of paths or, in the case when $r = s$ and $\mathcal{S}_{rs} = 2m_s$ when it is also necessary to merge (w_{m_s}, w_1), a single circuit. Excess values can then be added to one edge incident with each vertex of degree 2 in the paths, to prevent constraint violations. Inequality 15 shows that $\mathcal{E}_{rs}$ is sufficiently large to allow this.

Such a subgraph with $\mathcal{S}_{rs}$ edges is constructed for each pair r, s. Equation 15 guarantees that the union of these subgraphs, with the labelling of the vertices corresponding to each of the sites permuted if necessary, need have no vertices of degree greater than 2. Some vertices may have degree 0 or 1, but for each r exactly $\mathcal{S}_{rr}$ edges have to be added, each joining two vertices corresponding to cell $\mathcal{C}_r$. When adding these edges, the creation of small circuits should be avoided as far as possible. No excesses are necessary to prevent constraint violations involving these edges of weight c. When these final edges are added, each vertex has degree 2 and there are no constraint violations, given our initial assumptions. Thus a perfect 2-matching with no constraint violations is obtained.

2. **Use a heuristic to merge circuits without increasing span**
Suppose that the perfect 2-matching has more than one circuit, and that two of the circuits both contain a vertex from the same site $\mathcal{C}_r$. Then it is always possible to carry out a process of interchanging a pair of edges, each incident with a vertex from site $\mathcal{C}_r$, in such a way that the two circuits become one circuit. This process of interchange is as follows. Suppose that we have two edges (xv_i) and $(yv_j)$with $v_i, v_j \in \mathcal{C}_r$. By assumption $x \neq y$. Then we delete these two edges and insert edges (xv_j) and (yv_i). Note that in some cases it may be important that either two edges without excess are interchanged, or two edges with excess. This is continued until there are no pairs of circuits which both contain a vertex from the same site. No new constraint violations are created.

3. **Use a heuristic to merge circuits with a possible increase in span**
If the number of circuits has not been reduced to one, then circuits which do not each have a vertex from the same site can still be merged by deleting one edge from each circuit and inserting two new edges joining the resulting vertices of degree 1. However, additional constraint violations may arise. These can be removed by inserting additional excesses, so the span of the final assignment may increase.

4. **Construct an assignment from the circuit**
When a single Hamiltonian circuit in G_0' is obtained, it can be turned into a Hamiltonian path in G_0' by removing the dummy vertex v_0. The assignment is then obtained from the Hamiltonian path as described in section 14.

17 Conclusion

The main techniques for deriving lower bounds for minimum span frequency assignment have been surveyed. The techniques based on Mathematical Programming can now generally give computable and tight bounds for many of the practical problems that arise. For some problems the Perfect Two-Matching Bound is already tight. When it is not, the importance of the FAP constraints relative to the subtour elimination and integrality constraints has been noted. The remaining problem is to find general purpose algorithms to generate the best subgraph when a clique is not adequate, or when the constraint graph is too big for the maximum clique algorithm to be applied. In the former case, current methods require substantial manual input and even then are not always satisfactory.

In combination with the best current meta-heuristic algorithms this allows an exact or reasonably precise estimate of the necessary span for practical assignments without constraint violations. The ability to do this depends on the particular structures arising from the geometry of the transmitter locations. If the same techniques are applied to large randomly generated constraint graphs, the gap between the best of these lower bounds and the span of the best assignment known can be very large. It is fortunate that it is the practical problems for which the techniques are most effective.

Part III
Report Summary

This report has presented two significant contributions to the evaluation of the models and algorithms used in frequency assignment. The Benchmark Generator allows the generation of test data which is typical of frequency assignment problems. This is particularly useful when comparing representations of the FAP, but can be used to generate binary constraint data as well. The work on lower bounds has demonstrated that it is possible to produce extremely tight bounds for evaluating assignments in minimum span problems. Although these results are very satisfactory, there is still much work to be done in developing a method for determining the correct subgraph to which the bounds should be applied.

References

[1] Allen, S.M., Smith, D.H. and Hurley, S. (1999). Lower bounding techniques for frequency assignment, *Discrete Mathematics*, (to appear).

[2] Anderson, L.G. (1973). A simulation study of some dynamic channel assignment algorithms in a high capacity mobile telecommunications system. *IEEE Transactions on Communications*, **21**, 1294–1301.

[3] Carraghan, R. and Pardalos, P.M. (1990). An exact algorithm for the maximum clique problem. *Operations Research Letters*, **9**, 375–382.

[4] Castelino, D. and Stephens, N. (1998). A surrogate constraint tabu thresholding implementation for solving the frequency assignment problem. *Annals of Operations Research*, (to appear).

[5] Dunkin, N. and Allen, S. (1997). Frequency Assignment Problems: Representations and Solutions. *Internal Report, M-97-1, Division of Mathematics and Computing, University of Glamorgan and Department of Computer Science, Royal Holloway, University of London.*

[6] Gamst, A. (1986). Some lower bounds for a class of frequency assignment problems. *IEEE Transactions on Vehicular Technology*, **35**, 8–14.

[7] Gendreau, M., Soriano, P. and Salvail, L. (1993). Solving the maximum clique problem using a tabu search approach. *Annals of Operations Research*, **41**, 385–403.

[8] Hale, W.K. (1981). New spectrum management tools. *Proc. IEEE International Symposium on Electromagnetic Compatibility*, 47–53.

[9] Hale, W.K. (1980), Frequency assignment: theory and applications, *Proceedings of the IEEE*, **68**/12, 1487–1514.

[10] Hurley, S., Smith, D.H. and Thiel, S.U. (1997). FASoft: A system for discrete channel frequency assignment. *Radio Science*, **32**, 1921–1939.

[11] Janssen, J. and Kilakos, K. (1996). Polyhedral analysis of channel assignment problems: (I) tours. *CDAM Research Report, London School of Economics and Political Science CDAM-96-17.*

[12] Lanfear, T.A. (1989). Graph theory and radio frequency assignment. *NATO EMC Analysis Project No. 5, NATO Headquarters, 1110 Brussels.*

[13] Pekny, J.F. and Miller, D.L. (1994). A staged primal-dual algorithm for finding a minimum cost perfect two-matching in an undirected graph. *ORSA Journal on Computing*, **6(1)**, 68–81.

[14] Prim, R.C. (1957). Shortest connection networks and some generalizations. *Bell System Technical Journal*, **36**, 1389–1401.

[15] Raychaudhuri, A. (1985). Intersection assignments, T-colourings and powers of graphs. *Ph.D. Thesis, Rutgers University.*

[16] Sivarajan, K.N., McEliece, R.J. and Ketchum, J.W. (1989). Channel assignment in cellular radio. *Proceedings of the 39th Conference of the IEEE Vehicular Technology Society*, 846–859.

[17] Smith, D.H. and Hurley, S. (1997). Bounds for the frequency assignment problem. *Discrete Mathematics*, **167/168**, 571–582.

[18] Smith, D.H., Hurley, S. and Thiel, S.U. (1998). Improving heuristics for the frequency assignment problem. *European Journal of Operational Research*, **107**/1, 76–86.

[19] Smith, D.H., Hurley, S.,Thiel, S. and Allen, S.M. Benchmarks for the channel assignment problem. *IEEE Transactions on Vehicular Technology*, submitted.

[20] Tcha, D-w., Chung, Y-j. and Choi, T-j. (1997). A new lower bound for the frequency assignment problem. *IEEE/ACM Transactions on Networking*, **5**, 34–39.

[21] Thiel, S.U., Hurley, S. and Smith, D.H. (1997). The use of orderings in exact algorithms for the maximum clique problem. (submitted).

[22] Volgenant, A. (1990). Symmetric traveling salesman problems. *European Journal of Operational Research*, **49**, 153–154.

[23] Volgenant, T. and Jonker, R. (1982). A branch and bound algorithm for the symmetric traveling salesman problem based on the 1-tree relaxation. *European Journal of Operational Research*, **9**, 83–89.