3-1-2005

# Principles of Cost Minimisation in Wireless Networks

Vic Grout
*Glyndwr University*, v.grout@glyndwr.ac.uk



Recommended Citation
Grout,V. (2005)'Principles of Cost Minimisation in Wireless Networks',Journal of Heuristics, 11(2), 115 -133

# Principles of Cost Minimisation in Wireless Networks

**Abstract**
This paper considers variations of the minimum connected vertex cover problem to be found in the study of wireless network design. A simple, theoretic formulation is followed by a discussion of practical constraints. Two algorithms are given and results compared.

**Keywords**
Minimum connected vertex cover problem, Wireless networks

**Disciplines**
Computer Sciences

**Comments**
Original publication is available at www.springerlink.com

Journal of Heuristics, Vol. 11, No. 2, March 2005, pp115-133

# Principles of Cost Minimisation in Wireless Networks

**Vic Grout**

Centre for Applied Internet Research (CAIR), University of Wales, NEWI
Plas Coch Campus, Mold Road, Wrexham, LL11 2AW, UK
Tel: +44(0)1978 293203, Fax: +44(0)1978 293168
v.grout@newi.ac.uk

## *ABSTRACT*

*This paper considers variations of the minimum connected vertex cover problem to be found in the study of wireless network design. A simple, theoretic formulation is followed by a discussion of practical constraints. Two algorithms are given and results compared.*

## KEYWORDS

Minimum connected vertex cover problem, Wireless networks,
Node constraints, Edge constraints, Path constraints, Load constraints,
Add algorithm, Drop algorithm

## 1. INTRODUCTION

The *Minimum Connector Problem* (*MCP*) for cabled networks has been understood well for many years and applied in a variety of situations (Du & Pardalos, 1993). Given a number of *nodes*, or *vertices*, we seek to find the optimum set of edges (an *edge* is a link between two nodes) that fully connects the node set in question. (A network is connected if a *path* exists between each pair of nodes.) To this end, a *cost matrix* is applied to the nodes requiring interconnection with the cost element between each node pair - i.e. the cost of that edge - reflecting the expenditure, distance, difficulty, etc. involved in joining the two. Finding the *Minimum Spanning Tree* (*MST*) for the cost matrix will then result in the optimal solution across the nodes – i.e. the minimum cost set of connecting edges.

A typical example is given in Figure 1. Here, cost has been taken to be the Euclidean distance between node pairs and the MST minimises the total edge length in the connected solution. It should be noted that the *degree* of each node (the number of edges in the solution *adjacent* to that node – called the *valency* in some texts) differs from node to node. *Terminal* nodes (1, 3, 5, 9 & 10 in Figure 1) have degree *one*. A node of higher degree (2, 4, 6, 7 & 8) acts as some form of *connector* or *relay* between/among the two, or more, nodes to which it is connected.

In certain applications it may be necessary to apply constraints to the basic MST solution, reflecting practical restrictions/limitations. For example, in a traffic/load-carrying network, it may be necessary to limit the size of a link or relay. Simple and efficient techniques are known for finding the MST or for approximating it in its constrained form (Kershenbaum, 1993). The defining feature of the MCP/MST problem/solution is that the *objective function* is cost as given solely by the edge cost matrix. That is, the *optimization* process seeks to minimise the total cost

of the edges in the solution network, whereas no consideration is given to the degree of each/any node in the solution.

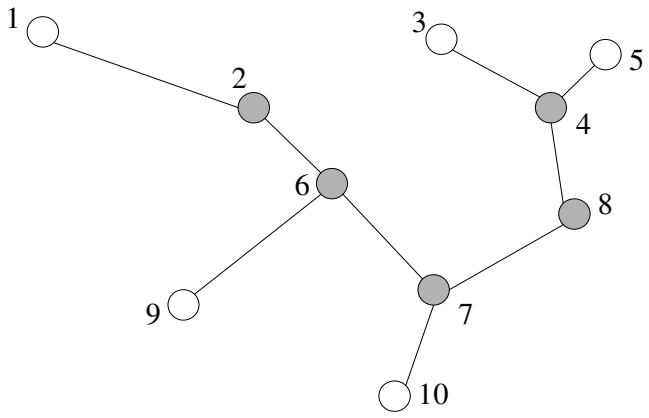



**Figure 1. Minimal Spanning Tree (F1)**

Many problems, but three in particular, have been noted in relation to this 'distance-only' approach to optimization. Firstly, the edge costs may not be known at the start of the optimization process. Secondly, a tree network is very vulnerable to failure. Thirdly, the cost of providing the necessary connection at the nodes themselves, rather than merely the edges between them, is ignored. The complete solution is a complex one (Grout, 1988). This paper, however, as an alternative, considers a variant of the MCP in which the cost of edges (except in the form of constraints – see later) is irrelevant. Instead, the cost is determined by the degree of the connecting nodes, reflecting the need for relay equipment to be installed at these vertices. The basis for this model is the graph-theoretic *minimum vertex cover* problem (Garey & Johnson, 1979) which, defined and constrained appropriately, proves to be particularly applicable to wireless applications across a wide range. The necessary concepts are introduced in stages, in line with increasingly demanding scenarios. The final formulation of this *Minimum Relay Problem* (*MRP*) is general enough to deal with most appropriate practical applications (Fowler, 2001 for example). Two approaches to the solution of the MRP are then considered and compared.

The minimum vertex cover problem has been considered elsewhere in relation to particular aspects of network design and management (Luo et al., 2002, for example) and in particular to specialised cost functions in wireless networks (Baldi et al., 2002, for example). However, this paper presents the first discussion of a *generalised* approach to wireless optimisation in a form that allows multiple constraint forms to be applied flexibly and independently or in combination.

## 2. THE UNCONSTRAINED MCP & MRP

Variants of the MRP are introduced by comparison to the MST.

### 2.1. Minimum Connector Problem (MCP)

In the conventional MCP, a *graph*, $\boldsymbol{G}$, is defined by a set of nodes (vertices), $\boldsymbol{V}$ and a set of edges, $\boldsymbol{E}$. If we can associate a cost, $c_e$ with each edge, $e \in \boldsymbol{E}$, then finding the MST for $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ will solve the MCP for the graph, $\boldsymbol{G}$. Denote this problem $P_{Con}$. The result will be a *tree*, $\boldsymbol{T}^*$, that minimises the *connection objective function*, $f_{Con}$, where

$$f_{Con}(T^*) = \sum_{e \in T^*} c_e = \min_T f_{Con}(T) = \min_T \sum_{e \in T} c_e \tag{1}$$

for all possible trees, $\boldsymbol{T} \subseteq \boldsymbol{E}$. An alternative (but equivalent and initially more useful) formulation is to define a cost matrix, $C = (c_{ij}: i,j \in \boldsymbol{V})$. ($1 \le i, j \le n$, where $n = |\boldsymbol{V}|$.) Then for $(i, j) \in \boldsymbol{E}$, $c_{ij}$ represents the cost of the link, $(i, j)$. For $(i, j) \notin \boldsymbol{E}$, $c_{ij} = \infty$. We also define a Boolean *link matrix*, $\Omega^{\boldsymbol{T}} = (\omega_{ij}^T : i,j \in \boldsymbol{V})$ as

$$\omega_{ij}^T = \begin{cases} 1 : (i, j) \in T \\ 0 : (i, j) \notin T \end{cases} \tag{2}$$

denoting whether an edge is present in any given solution, $\boldsymbol{T}$. The MST, $\boldsymbol{T}^*$, is then found for the matrix, $C$, giving $\Omega^{\boldsymbol{T}^*}$. Any $\boldsymbol{T}$, and in particular $\boldsymbol{T}^*$, will have $n-1$ edges and

$$f_{Con}(T^*) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} \omega_{ij}^{T^*} = \min_T f_{Con}(T) = \min_T \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} \omega_{ij}^T . \tag{3}$$

(Assuming $C$ to be symmetric about the leading diagonal, the elements on and below the leading diagonal may be ignored.) The two solutions, (1) and (3), will be identical.

### 2.2. Minimum Relay Problem (MRP)

In a wireless network however, if a link is viable at all, then there being no expense involved in cabling, assigning a cost to the equivalent edge is inappropriate. Instead, the true cost of the network is derived from the cost of the relays at the connecting nodes. In Figure 1, for example, the number of relays is five. The problem may be restated accordingly.

As before, define $n$ nodes: $1, 2, ..., n$ by the set $\boldsymbol{V}$. $n = |\boldsymbol{V}|$. Suppose initially that an edge is *feasible* between any pair of nodes. $\boldsymbol{E} = \boldsymbol{V} \times \boldsymbol{V}$, the Cartesian product of $\boldsymbol{V}$ with itself ($\{(i, j): i, j \in \boldsymbol{V}\}$). Define a (*connected*) *network*, $\boldsymbol{N}$, to be any set of edges, $\boldsymbol{N} \subseteq \boldsymbol{E}$ in which (at least) one path exists between any pair of nodes, $i, j \in \boldsymbol{V}$.

As with a tree structure, a network, $\boldsymbol{N}$, may be defined by the link matrix, $\Omega^{\boldsymbol{N}} = (\omega_{ij}^N : i,j \in \boldsymbol{V})$ as

$$\omega_{ij}^N = \begin{cases} 1 : (i, j) \in N \\ 0 : (i, j) \notin N \end{cases} . \tag{4}$$

However, there being no cost matrix to consider, $\Omega^N$ in isolation has no significance. Instead, it defines in turn a *relay vector*, $\underline{\sigma}^N = (\sigma_i^N : i \in \boldsymbol{V})$, where

$$\sigma_i^N = \begin{cases} 1 : \sum_{j=1}^{n} \omega_{ij}^N > 1 \\ 0 : \sum_{j=1}^{n} \omega_{ij}^N = 1 \end{cases}. \tag{5}$$

$\sigma_i^N$ defines whether node *i* is a relay in the network, $\boldsymbol{N}$. For the basic MRP, we seek to find the network, $\boldsymbol{N}^*$, that minimises the total number of relays, i.e. such that, for the *relay objective function*, $f_{Rel}$,

$$f_{Rel}(N^*) = \sum_{i=1}^{n} \sigma_i^{N^*} = \min_N f_{Rel}(N) = \min_N \sum_{i=1}^{n} \sigma_i^N \tag{6}$$

for all (connected) networks, $\boldsymbol{N}$.

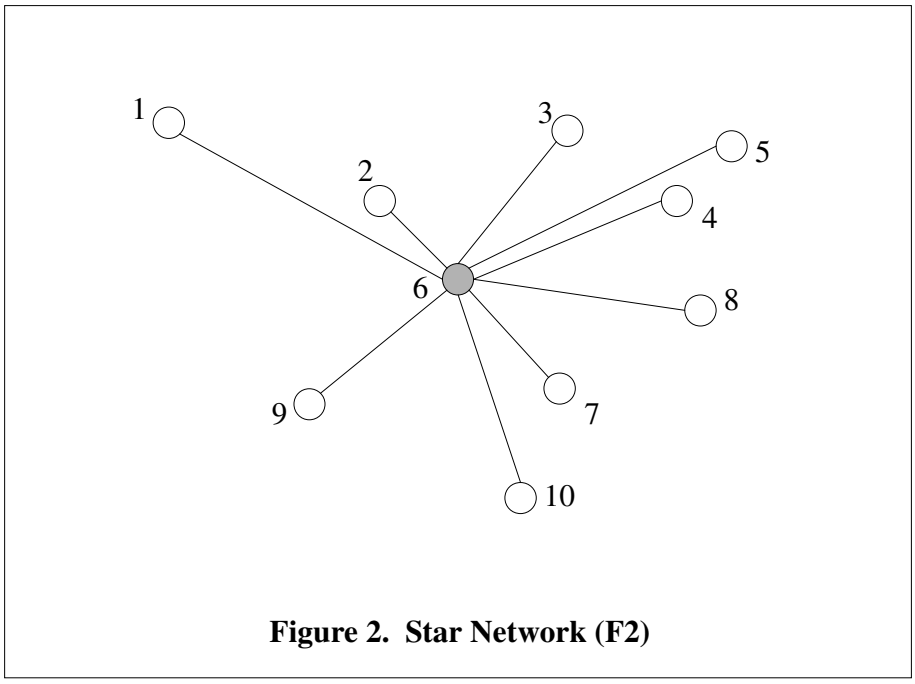



**Figure 2. Star Network (F2)**

LEMMA *For a graph, $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, with $n = |\boldsymbol{V}| > 2$, there are n networks $\boldsymbol{N} \subseteq \boldsymbol{E}$, with $f_{Rel}(\boldsymbol{N}) = 1$.*

PROOF

If $\boldsymbol{N}_{<h>}$ ($h \in \boldsymbol{V}$) is the star network (as in Figure 2) with *h* as its *hub*, then $\Omega^{N_{<h>}}$ is given by

$$\omega_{ij}^{N_{<h>}} = \begin{cases} 1 : i = h \vee j = h \\ 0 : i \neq h \wedge j \neq h \end{cases}, \tag{7}$$

and, $\underline{\sigma}^{N_{<h>}}$, in turn, by

$$\sigma_i^{N_{<h>}} = \begin{cases} 1 : i = h \\ 0 : i \neq h \end{cases}. \tag{8}$$

Then $f_{\mathrm{Re}l}(N_{<h>}) = \sum_{i=1}^{n} \sigma_i^{N_{<h>}} = 1$ and there are $n$ such networks, $h = 1, 2, .., n$.

□

Consequently the solution to the MRP, denoted $P_{Rel}$, is trivial. For $n > 2$, there must be at least one relay in the network (at least one node must be adjacent to at least two others) and any *star* network of the form shown in Figure 2 (with one relay) will be optimal. The problem only takes interesting form when constrained.

## 2.3. Minimum Degree Relay Problem (MDRP)

The formulation given in Section 2.2 will be valid for a wireless network in which transmitters and receivers (*transceivers*) are *omni-directional*. That is, if a single piece of equipment will suffice to maintain the link with all adjacent nodes. If this is not the case, if a separate (*steered*) transceiver is required for each link, then it is necessary to consider the degree of a relay as a measurement of its cost. In fact, this equipment may be necessary at terminal nodes also. In this simple case, the cost of a network is given by its total degree, the sum of the degrees of all nodes.

With $\Omega^{N}$ defined as before for a network, $\boldsymbol{N}$, the *network degree vector*, $\underline{\delta}^{N} = (\delta_i^{N} : i \in \boldsymbol{V})$, is defined as

$$\delta_i^{N} = \sum_{j=1}^{n} \omega_{ij}^{N} \tag{9}$$

and the *node degree objective function*, $f_{nDg}$, for the problem $P_{nDg}$, as

$$f_{nDg}(N^{*}) = \sum_{i=1}^{n} \delta_i^{N^{*}} = \min_{N} f_{nDg}(N) = \min_{N} \sum_{i=1}^{n} \delta_i^{N}. \tag{10}$$

LEMMA *For a graph, $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, with $n = |\boldsymbol{V}| > 2$, for any tree, $\boldsymbol{T} \subseteq \boldsymbol{E}$, $f_{nDg}(\boldsymbol{T}) = 2(n - 1) = \min_{N} f_{nDg}(\boldsymbol{N})$ for all networks, $\boldsymbol{N} \subseteq \boldsymbol{E}$.*

PROOF

$\boldsymbol{T}$ will have $n-1$ edges, each of which contributes one to the degree of each of its end nodes. Thus $f_{nDg}(\boldsymbol{T}) = 2(n-1)$. For any network, $\boldsymbol{N} = \boldsymbol{T} \cup \boldsymbol{B}$ ($\boldsymbol{B} \subseteq \boldsymbol{E} - \boldsymbol{T}$), $f_{nDg}(\boldsymbol{N}) = f_{nDg}(\boldsymbol{T}) + 2|\boldsymbol{B}| = 2(n-1) + 2|\boldsymbol{B}| \geq 2(n-1)$.

□

So $f_{nDg}$, in its unconstrained form, has constant (minimum) value, $f_{nDg}(\boldsymbol{T}) = 2(n - 1)$, for all trees, $\boldsymbol{T}$. Ignoring transceiver costs at terminal nodes (which may be an integral part of the basic equipment), gives the *relay degree objective function*, $f_{rDg}$, for the problem $P_{rDg}$ as

$$f_{rDg}(N^*) = \sum_{i=1}^{n} \sigma_i^{N^*} \delta_i^{N^*} = \min_N f_{rDg}(N) = \min_N \sum_{i=1}^{n} \sigma_i^N \delta_i^N , \qquad (11)$$

minimising the sum of the degrees of all *relay* nodes.

LEMMA *For a graph, $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, with $n = |\boldsymbol{V}| > 2$, for any tree, $\boldsymbol{T} \subseteq \boldsymbol{E}$ with $r$ relays, $f_{rDg}(\boldsymbol{T}) = n + r - 2$.*

PROOF

$f_{nDg}(\boldsymbol{T}) = 2(n-1)$. There are $n - r$ non-relay nodes, each of degree *one*. So $f_{rDg}(\boldsymbol{T}) = 2(n-1) - (n-r) = n + r - 2$.

□

So, for a tree, $\boldsymbol{T}$, of $n$ nodes and $r$ relays, $f_{rDg}(\boldsymbol{T}) = n + r - 2$, which is minimised once again by any network of the form in Figure 2 ($n + r - 2$, for fixed $n$, is minimised by minimising $r$). Again, the (MDRP) problem in its unconstrained form is trivial.

## 3. THE CONSTRAINED MRP

There are a number of constraints that may be applied to the MRP ($P_{Rel}$) or MDRP ($P_{nDg}$ or $P_{rDg}$). These relate broadly to practical restrictions such as invalid link/relay choices, redundancy/robustness requirements, maximum capacities and dealing with equipment already in place. This section considers each in turn.

### 3.1. Edge Constraints

There are two reasons, in particular, why an edge between two nodes may not be feasible. Both depend explicitly upon the technology concerned but may be generalised. Firstly, the distance between nodes may be too great. Secondly, many forms of wireless link require *line-of-sight* (*l-o-s*) adjacency, which may or may not be available.

Define the *distance matrix*, $D = (d_{ij}: i,j \in \boldsymbol{V})$ where $d_{ij}$ is the distance between nodes $i$ and $j$. $D$ is partially analogous to the cost matrix, $C$. In situations obeying the two-dimensional Euclidean model, $d_{ij} = [(x_i-x_j)^2 + (y_i-y_j)^2]^{1/2}$, where $(x_i, y_i)$ and $(x_j, y_j)$ are the Cartesian co-ordinates of nodes $i$ and $j$. However, non-Euclidean distances permit local factors to be considered. Let the maximum link distance be $d_{max}$. Define the Boolean *line-of-sight* (*l-o-s*) *matrix*, $\Pi = (\pi_{ij}: i,j \in \boldsymbol{V})$ by $\pi_{ij} = 1$ if line-of-sight exists between nodes $i$ and $j$, and $\pi_{ij} = 0$ otherwise.

The *edge viability matrix*, $Z = (z_{ij}: i,j \in \boldsymbol{V})$ is then defined as

$$z_{ij} = \begin{cases} 1 : d_{ij} \le d_{\max} \,\&\, \pi_{ij} = 1 \\ 0 : otherwise \end{cases}, \tag{12}$$

which, in turn, redefines the edge set, $\boldsymbol{E}$, as $(i, j) \in \boldsymbol{E} \Leftrightarrow z_{ij} = 1$. The problem is then to find a network, $\boldsymbol{N}$, that minimises the objective function, $f_{Rel}$, $f_{nDg}$ or $f_{rDg}$ for the graph, $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$. $P_{nDg}$ remains trivial since any tree will minimise $f_{nDg}$. However, $P_{Rel}$ and $P_{rDg}$ correspond to the NP-complete (Connected) Vertex Cover and Maximum Leaf Spanning Tree problems (Garey & Johnson, 1979). Neither is trivial and solutions are discussed in Section 4. Before this, further constraints and complexities are considered.

## 3.2. Node Constraints

The installation of relay equipment at nodes may have pre-requisites. The site must be technologically suitable and, in the case of a (e.g. broadband) subscriber distribution network, permission will be required. The absence of either will make a node infeasible in a similar manner to edges in the previous sub-section.

A *node viability vector*, $\underline{z} = (z_i : i \in \boldsymbol{V})$, can be defined as $z_i = 1$ if a relay is permitted at node $i$ and $z_i = 0$ otherwise. This Boolean vector may be extended, if necessary, to a generalised natural number equivalent, $\underline{z} = (z_i = \lambda_i : i \in \boldsymbol{V})$, where $\lambda_i$ is the maximum degree permitted at node $i$. Alternatively, it may be known that key equipment already exists at certain locations, in which case it will be appropriate to constrain a relay to a particular node (a *fixed relay*). Whilst reducing the size of the solution space and hence (possibly) the time complexity of solution, these additional constraints may increase the design complexity (space complexity) of the solving algorithm.

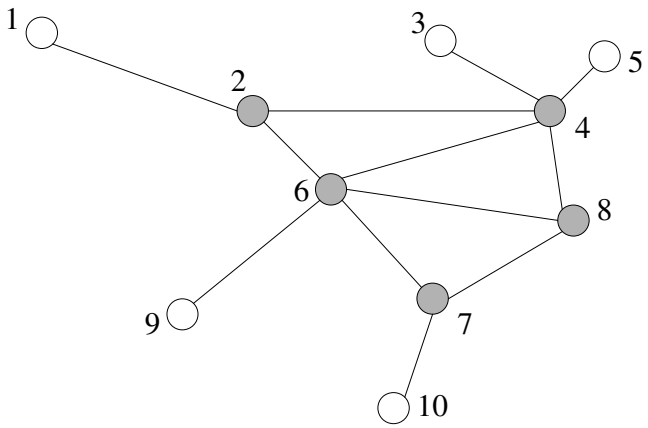



**Figure 3. Multiple Paths/Circuits (F3)**

### 3.3. Path Constraints

In general, for any graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, for any given tree, $\boldsymbol{T}$, there will be a number of connected networks, $\boldsymbol{N}$, such that $\boldsymbol{T} \subseteq \boldsymbol{N}$. Comparing the networks in Figures 1, 3 & 4, $f_{Rel}(F1) = f_{Rel}(F3) < f_{Rel}(F4)$, $f_{nDg}(F1) < f_{nDg}(F3) < f_{nDg}(F4)$ and $f_{rDg}(F1) < f_{rDg}(F3) < f_{rDg}(F4)$.

Although clearly sub-optimal with respect to the restrictions of subsections 3.1 and 3.2 (other than the last, fixed-relay constraint), there are two types of constraint that may justify solutions of the form shown in Figures 3 & 4. Both address the reliability of the solution network. Firstly, it may be necessary that the path between two given nodes be no longer than some maximum number of edges. Secondly, it may be required that a given number of independent paths exist between the two nodes. Paths may be *node-* or *edge-independent*, giving two separate constraint forms.

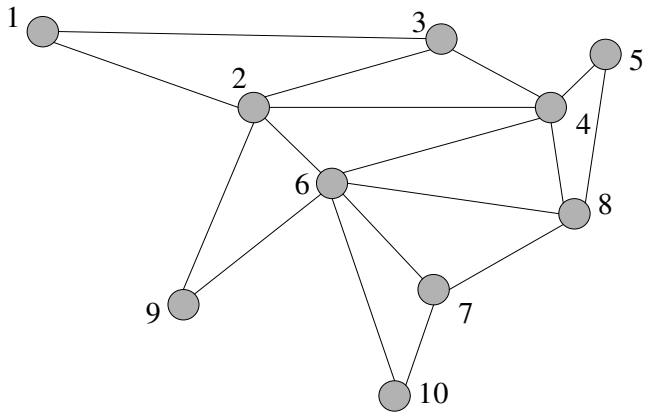



**Figure 4. Multiple Paths/Circuits (F4)**

Define the *path length matrix*, $P = (p_{ij}: i,j \in \boldsymbol{V})$ to be such that $p_{ij}$ represents the maximum number of links in the shortest path between $i$ and $j$. If $p_{ij} = 1$, for example, then $i$ and $j$ are constrained to be directly connected. Shortest paths are easily computed (Dijkstra, 1959). The problem of establishing multiple bounded-length paths, however, is NP-hard (Garey & Johnson, 1979) and thus unusable as a constraint. A more practical alternative, assuming some form of dynamic routing capability, is to specify a minimum degree at key nodes. This may be achieved in conjunction with the constraints of the previous sub-section or explicitly through the definition of the *minimum degree vector*, $\underline{\rho} = (\rho_i: i \in \boldsymbol{V})$ where $\rho_i$ gives the minimum number of nodes to which $i$ must be connected.

Applying path constraints will increase the complexities of problems $P_{Rel}$, $P_{nDg}$ and $P_{rDg}$, except in the extremely constrained form of Figure 4 in which $P_{Rel}(F4) = n$ (constant).

### 3.4. Load Constraints

Load constraints are based upon a knowledge of the projected traffic in the network. This is not the same as defining static edge costs since the load on a given link or node will depend upon the final topology of the network. As this is indeterminate at the outset, it may not serve as input to the optimization process. Instead, an (initial) *traffic matrix*, $T = (t_{ij}: i,j \in \boldsymbol{V})$, is defined with $t_{ij}$ giving the traffic originating at node $i$ and destined for $j$. This figure is independent of the path the traffic may take. The total traffic between $i$ and $j$ is then $t_{ij} + t_{ji}$.

In a solution network, $\boldsymbol{N}$, define the *load matrix*, $L^N = (l_{ij}^N : i,j \in \boldsymbol{V})$ where $l_{ij}^N$ represents the traffic carried on the link *(i, j)* (in the direction $i$ to $j$) in the configuration $\boldsymbol{N}$. Loads can be difficult to calculate across an entire network but can be determined individually as follows. For a fully (edge) feasible network, the load on each *(i, j)* link is the traffic, $t_{ij}$, between the two nodes. For *(i, j)* with $z_{ij} = 0$, a shortest path, using feasible edges, can be found (Dijkstra, 1959) and $t_{ij}$ added to each link in the path. If a link is removed, as in the algorithm in subsection 4.2, then load is recalculated similarly.

Define also a *load limit matrix*, $K = (\kappa_{ij}: i,j \in \boldsymbol{V})$, and a *load limit vector*, $\underline{\kappa} = (\kappa_i: i \in \boldsymbol{V})$. $\kappa_{ij}$ gives the maximum traffic permitted on the link *(i, j)*, and $\kappa_i$ through node $i$. Then, for any valid solution network, $\boldsymbol{N}$,

$$\xi_{ij} = l_{ij}^N + l_{ji}^N \leq \kappa_{ij} \qquad \text{for all } i,j \in \boldsymbol{V} \tag{13}$$

and

$$\zeta_i = \sum_{j=1}^{n} (l_{ij}^N + l_{ji}^N) \leq \kappa_i \qquad \text{for all } i \in \boldsymbol{V}\,. \tag{14}$$

## 4. ALGORITHMS

We begin this section by counting solutions.

LEMMA *For a set of vertices $\boldsymbol{V}$ with $n = |\boldsymbol{V}| > 2$, there are $2^{n(n-1)/2}$ possible graphs, $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$.*

PROOF

Each node, *1, 2, .., n*, may be connected to *n-1* others, giving *n(n-1)* possible edges, counting each edge twice, so *n(n-1)/2* in truth. Each of these may or may not be a member of $\boldsymbol{E}$, giving $2^{n(n-1)/2}$ possible combinations in all.

□

LEMMA *For a set of vertices $\boldsymbol{V}$ with $n = |\boldsymbol{V}| > 2$, there are $n^{n-2}$ possible trees, $\boldsymbol{T}$.*

PROOF

There are numerous proofs of this famous result. See Moon (1970) for a discussion.

□

THEOREM *For a set of vertices* $\mathbf{V}$ *with* $n = |\mathbf{V}| > 2$, $\Delta_n$, *the number of connected networks is given recursively by*

$$\Delta_n = 2^{n(n-1)/2} - \sum_{\mu=1}^{n-1} \frac{(n-1)!\Delta_\mu 2^{(n-\mu)(n-\mu-1)/2}}{(\mu-1)!(n-\mu)!}. \tag{15}$$

PROOF
(Adapted from Harary & Palmer, 1973).

If $N_n$ is the number of (connected or disconnected) networks on $n$ nodes then

$$N_n = 2^{n(n-1)/2}. \tag{16}$$

Let $\Delta_n$ be the number of connected networks on $n$ nodes and $R_n$ be the number of *rooted* networks on $n$ nodes. (A rooted network is a (connected or disconnected) network with one particular node, the *root*, distinguished from the rest.) For each network of $n$ nodes, there are $n$ such rooted networks so

$$R_n = n N_n. \tag{17}$$

Finally, in preparation, let $R_{n(\mu)}$ be the number of rooted networks on $n$ nodes whose root lies in a connected component of size $\mu$. The number of ways of selecting $\mu$ nodes from $n$ is

$${}^nC_\mu = \frac{n!}{\mu!(n-\mu)!}. \tag{18}$$

There are $\Delta_\mu$ connected networks on $\mu$ nodes and $N_{n-\mu}$ (connected or disconnected) networks on the $n$-$\mu$ nodes that remain. This gives

$$R_{n(\mu)} = \mu\left(\frac{n!}{\mu!(n-\mu)!}\right)\Delta_\mu N_{n-\mu} = \frac{n!\Delta_\mu N_{n-\mu}}{(\mu-1)!(n-\mu)!}. \tag{19}$$

Combining these expressions, a network (of size $n$) with a connected component of size $n$ is trivially a connected network so that

$$R_{n(n)} = n\Delta_n \tag{20}$$

giving

$$\begin{aligned}\Delta_n &= \frac{R_{n(n)}}{n} = \frac{1}{n}\left(R_n - \sum_{\mu=1}^{n-1} R_{n(\mu)}\right) = N_n - \frac{1}{n}\sum_{\mu=1}^{n-1}\frac{n!\Delta_\mu N_{n-\mu}}{(\mu-1)!(n-\mu)!} \\ &= 2^{n(n-1)/2} - \sum_{\mu=1}^{n-1}\frac{(n-1)!\Delta_\mu 2^{(n-\mu)(n-\mu-1)/2}}{(\mu-1)!(n-\mu)!}\end{aligned} \tag{21}$$

□

Constraints restrict the solution space somewhat but all of these counts increase exponentially with $n$. Optimization by exhaustive search is not a viable option and an MST solution, minimising total distance rather than node degree, will give poor results (Section 5). As an alternative, this section offers two simple, greedy algorithms that, in practice, work well.

Given a graph, $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ and a valid connected solution, $\boldsymbol{N}$, the constraints of Section 3 may be partitioned according to the point at which they take effect. Define the edge and node constraints from sub-sections 3.1 and 3.2, apart from the fixed relay constraint, as the *add constraints*. Define the fixed relay constraint and the performance constraints from sub-sections 3.3 and 3.4 as the *drop constraints*. Adding a new link or relay to $\boldsymbol{N}$ may only violate an add constraint, not a drop constraint. (It is not permitted to add a link $(i, j)$ for which $z_{ij}=0$ or a relay at $i$ where $z_i=0$.) Removing a link or relay from $\boldsymbol{N}$ may only violate a drop constraint, not an add constraint. This observation suggests adaptations of two standard algorithms (Kershenbaum & Chou, 1974).

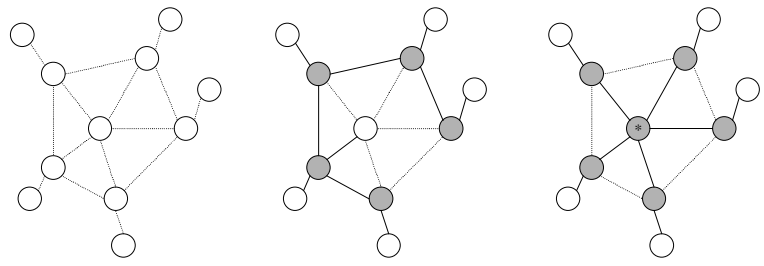
## 4.1. The Add Algorithm

For a graph, $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, define the *edge matrix*, $E = (e_{ij}: i, j \in \boldsymbol{V})$ for $\boldsymbol{G}$ as

$$e_{ij} = \begin{cases} 1 : (i, j) \in E \\ 0 : (i, j) \notin E \end{cases} \tag{22}$$

and the *valency vector*, $\underline{v} = (v_i: i \in \boldsymbol{V})$, for $\boldsymbol{G}$ as

$$v_i = \sum_{j=1}^{n} e_{ij} \,. \tag{23}$$



(a) Viable edges (b) Optimal solution (c) Greedy solution (node * selected first in Add algorithm)

**Figure 5. Greedy Comparison**

A basic heuristic for the vertex cover problem is given by Papadimitriou & Steiglitz (1998). However, this must be extended to take constraints, particularly connectivity, into account. If only add constraints are to be applied to the problem, then the following algorithm will approximate an optimal solution. It constructs a network, $\boldsymbol{N}$, from an empty link set, using the *temporary spanning vector*, $\underline{s}^N = ( s_i^N : i \in \boldsymbol{V})$, where $s_i^N = 0$ initially for all $i \in \boldsymbol{V}$ and $s_i^N = 1$ as $i$ is included.

**ADD**
{ Initialization }
**for all** $i \in \boldsymbol{V}$ **do**
  $s_i^{\boldsymbol{N}} = 0$
**for all** $i, j \in \boldsymbol{V}$ **do**
  $\omega_{ij}^{\boldsymbol{N}} = 0$
**find** $i$ **such that**
  $v_i = \max_j v_j$
$s_i^{\boldsymbol{N}} = 1$
{ Growth }
**while there exists** $j$ **such that**
  $s_j^{\boldsymbol{N}} = 0$ **do** {
  **for all** $j \in \boldsymbol{V}$ **such that**
    $z_j = 1$ **and** $e_{ij} = 1$ **and** $s_j^{\boldsymbol{N}} = 0$ **do** {
    $\omega_{ij}^{\boldsymbol{N}} = 1$
    $s_j^{\boldsymbol{N}} = 1$ }
  **find** $i$ **such that**
    $v_i - \delta_i^{\boldsymbol{N}} = \max_j (v_j - \delta_j^{\boldsymbol{N}})$ **where** $s_j^{\boldsymbol{N}} = 1$ }

A *spanning relay* is chosen initially as the node of highest degree. A link is then established between it and all adjacent nodes. From the nodes currently spanned, a new spanning relay is selected, adjacent to the maximum number of unspanned nodes, and the process is repeated. This is a node-based, constrained, maximising form of Prim's algorithm (Prim, 1957) although, in this case, the greedy algorithm is not exact (Figure 5). It does, however, perform well in practice (Section 5).

## 4.2. The Drop Algorithm

The drop constraints of Section 3 require a different approach. The *Drop algorithm* works in reverse and can be applied with add and drop constraints or drop constraints only. An initial solution is a network, fully connected so far as add constraints permit, from which links, and consequently relays, are removed, subject to drop constraints.

**DROP**
{ Initialization }
**for all** $i, j \in \boldsymbol{V}$ **such that**
  $(z_i = 1$ *or* $z_j = 1)$ **and** $z_{ij} = 1$ **do**
  $\omega_{ij}^{\boldsymbol{N}} = 1$
{ Reduction }
**while there exists** i, $j$ **such that**
  $\delta_i^{\boldsymbol{N}} > \rho_i$ **and** $\delta_j^{\boldsymbol{N}} > \rho_j$ **do** {
  **find** $i, j$ **such that**
    <**Can_remove**$(i, j)$> **and** $\delta_i^{\boldsymbol{N}} - \rho_i = \min_k (\delta_k^{\boldsymbol{N}} - \rho_k)$
  $\omega_{ij}^{\boldsymbol{N}} = 0$ }

The Boolean function, **Can_remove**(i,j), is defined as:

**Can_remove**(*i,j*) =
  **there exists** *k* **such that**
    $\omega_{ik}^{N} = 1$ **and** $\omega_{jk}^{N} = 1$ **and**
    $\xi_{ik} + \xi_{ij} \leq \kappa_{ik}$ **and** $\xi_{jk} + \xi_{ij} \leq \kappa_{jk}$ **and** $\zeta_k + \xi_{ij} \leq \kappa_k$

**Can_remove**(i,j) finds a relay *k* through which to route traffic currently on *(i,j)*, subject to satisfying the load constraints. Satisfying path constraints is implicit in the min operation that follows. Once again, the Drop algorithm does not guarantee optimality but performs well in practice.

## 5. COMPARISONS

For a (fully-feasible) graph of just *9* nodes, there are approximately *5 million* trees ($n^{n-2}$) and *66 billion* connected networks (Equation 15). For larger numbers of nodes, the ideal of comparing these heuristics with results from exhaustive search optimization is an impractical one. Instead, the results produced by both algorithms, and their run times, are compared with a simple MST process as well as each other. The MST should not be expected to provide a good solution but it provides a benchmark against which to compare other methods.

### 5.1. Accuracy

Test instances were generated randomly in the unit square, with *x*- and *y*- co-ordinates uniformly, independently distributed in the interval *[0, 1]*. Numbers of test runs from *20* to *100* were used with a variety of constraint combinations and the MST, Add and Drop algorithms compared for each. Some instances (disconnected graphs, for example) were infeasible and not included in summary results.

Tested constraints consisted of the following, applied individually and in combination:

- Maximum link distance ($d_{max}$ from section 3.1) of between *0.1* and *1.0* of the unit square containing all co-ordinates. (Shown as *T*, the transmit limit in Table 1 for values between *T = 0.1* and *T = 1.0.*)
- Line-of-sight $\pi_{ij}$ for each pair *(i,j)*, generated at random with probability *L*, with values between *L = 0.1* and *L = 1.0.*
- Node viabilities $z_i$ generated randomly with probability in the range *0.5* to *1.0.* (Again, smaller values almost always give infeasible graphs.)
- Minimum degrees $\rho_i$ randomly assigned from values *1*, *2*, *3* and *4*. (Higher values generally give infeasible solutions.)
- Traffic matrices *T* with elements $t_{ij}$ randomly generated on the interval *[0,1]* with load limits $\kappa_i$ and $\kappa_{ij}$ randomly generated on the interval *[0,n]*, (the load limit on a particular link or node/relay being anything up to a maximum single link load multipled by the number of nodes.)

Figures 6 to 10 show typical results. Figure 6 shows an initial feasible edge set generated on *200* nodes with complete l-o-s connectivity, a transmission distance limit of *0.15* (of unit distance) and no node constraints.

**Table 1. Comparing Add and Drop Performance with MST**

L: l-o-s probability - used to generate individual line-of-sight values

T: transmit limit ($d_{max}$) as a proportion of the unit square containing all nodes

A: Add algorithm D: Drop algorithm

Rel: f(Rel) for Add or Drop in proportion to MST
nDg: f(nDg) for Add or Drop in proportion to MST
rDg: f(rDg) for Add or Drop in proportion to MST

No. of nodes: 30 No. of runs: 100

| | T = 0.5 | | T = 1.0 | |
|---|---|---|---|---|
| L = 0.1 | A/Rel:#### | D/Rel:#### | A/Rel:0.66 | D/Rel:0.69 |
| | A/rDg:#### | D/rDg:#### | A/rDg:0.87 | D/nDg:0.89 |
| L = 0.5 | A/Rel:0.33 | D/Rel:0.34 | A/Rel:0.16 | D/Rel:0.18 |
| | A/rDg:0.73 | D/rDg:0.74 | A/rDg:0.67 | D/nDg:0.69 |
| L = 1.0 | A/Rel:0.14 | D/Rel:0.16 | A/Rel:0.05 | D/Rel:0.06 |
| | A/rDg:0.62 | D/rDg:0.65 | A/rDg:0.58 | D/nDg:0.60 |

No. of nodes: 100 No. of runs: 50

| | T = 0.5 | | T = 1.0 | |
|---|---|---|---|---|
| L = 0.1 | A/Rel:0.45 | D/Rel:0.48 | A/Rel:0.27 | D/Rel:0.29 |
| | A/rDg:0.79 | D/rDg:0.80 | A/rDg:0.73 | D/nDg:0.75 |
| L = 0.5 | A/Rel:0.12 | D/Rel:0.13 | A/Rel:0.06 | D/Rel:0.07 |
| | A/rDg:0.65 | D/rDg:0.66 | A/rDg:0.62 | D/nDg:0.67 |
| L = 1.0 | A/Rel:0.05 | D/Rel:0.07 | A/Rel:0.01 | D/Rel:0.02 |
| | A/rDg:0.58 | D/rDg:0.60 | A/rDg:0.57 | D/nDg:0.62 |

No. of nodes: 300 No. of runs: 30

| | T = 0.1 | | T = 1.0 | |
|---|---|---|---|---|
| L = 0.1 | A/Rel:#### | D/Rel:#### | A/Rel:0.11 | D/Rel:0.15 |
| | A/rDg:#### | D/rDg:#### | A/rDg:0.66 | D/nDg:0.72 |
| L = 0.5 | A/Rel:0.56 | D/Rel:0.59 | A/Rel:0.03 | D/Rel:0.03 |
| | A/rDg:0.81 | D/rDg:0.85 | A/rDg:0.62 | D/nDg:0.67 |
| L = 1.0 | A/Rel:0.31 | D/Rel:0.35 | A/Rel:.004 | D/Rel:.006 |
| | A/rDg:0.71 | D/rDg:0.74 | A/rDg:0.57 | D/nDg:0.61 |

No. of nodes: 1000 No. of runs: 20

| | T = 0.1 | | T = 1.0 | |
|---|---|---|---|---|
| L = 0.1 | A/Rel:#### | D/Rel:#### | A/Rel:0.04 | D/Rel:0.07 |
| | A/rDg:#### | D/rDg:#### | A/rDg:0.65 | D/nDg:0.71 |
| L = 0.5 | A/Rel:0.19 | D/Rel:0.20 | A/Rel:.009 | D/Rel:.016 |
| | A/rDg:0.68 | D/rDg:0.75 | A/rDg:0.60 | D/nDg:0.66 |
| L = 1.0 | A/Rel:0.09 | D/Rel:0.13 | A/Rel:.001 | D/Rel:.002 |
| | A/rDg:0.60 | D/rDg:0.67 | A/rDg:0.56 | D/nDg:0.61 |

#### = no feasible solutions in test instances

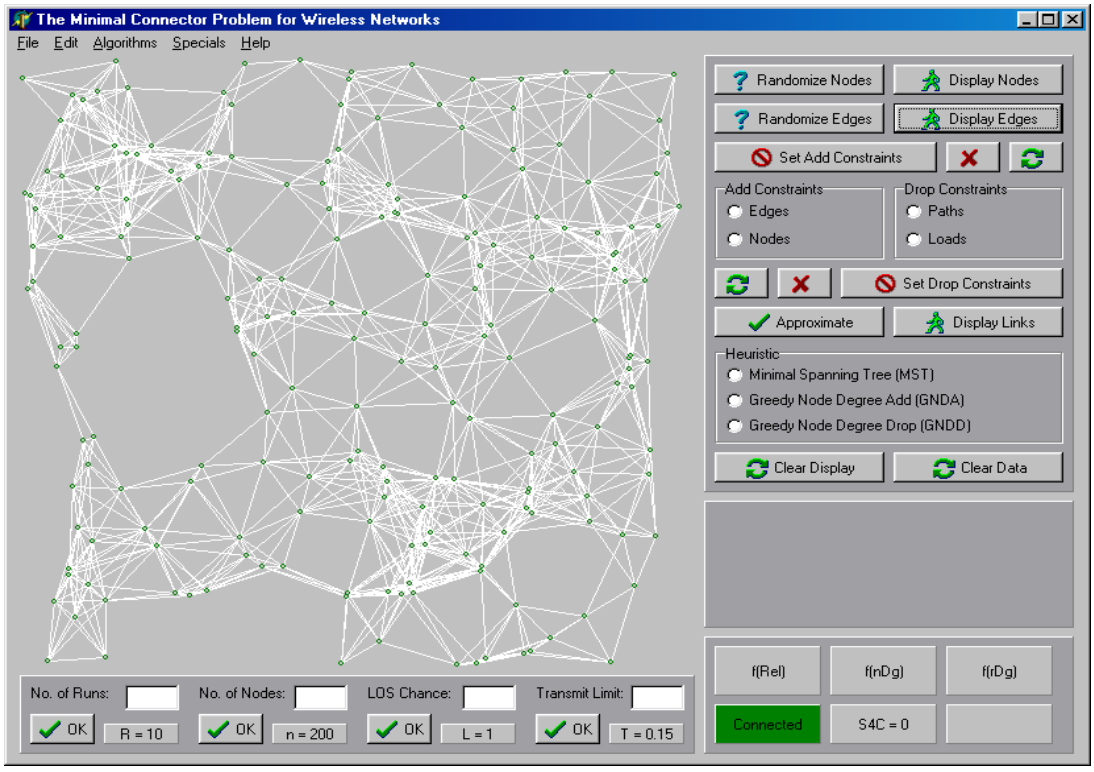



**Figure 6. Feasible Edges**

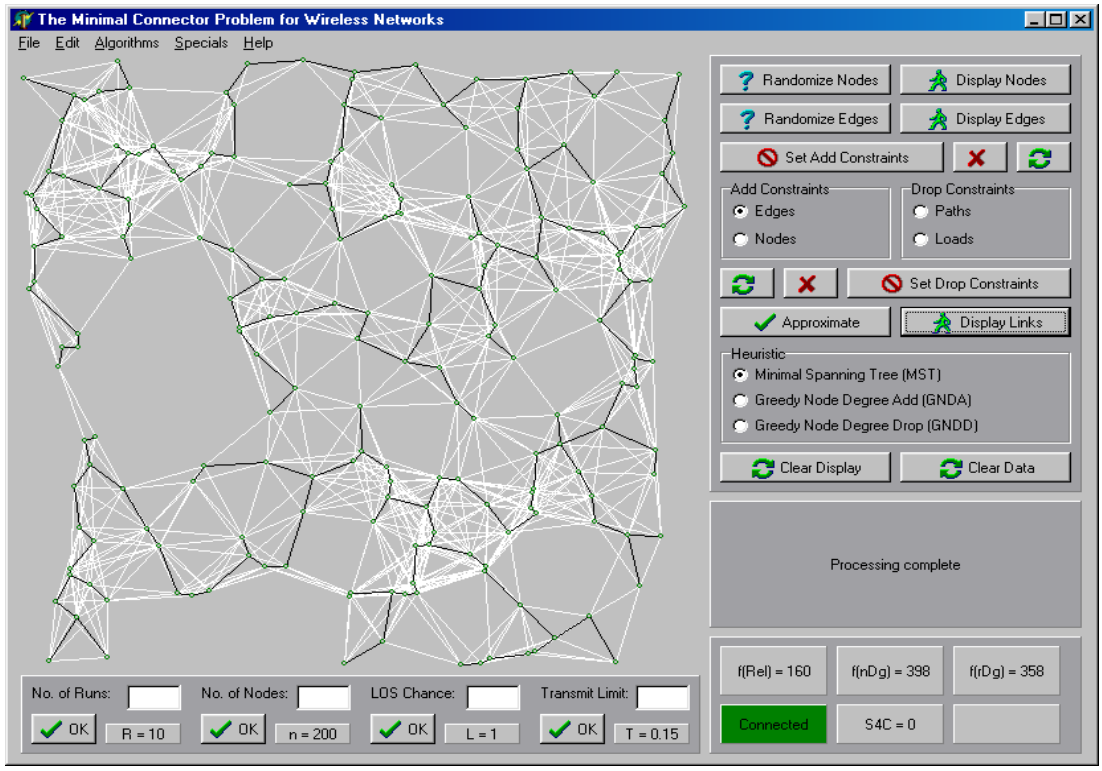



**Figure 7. MST Solution**

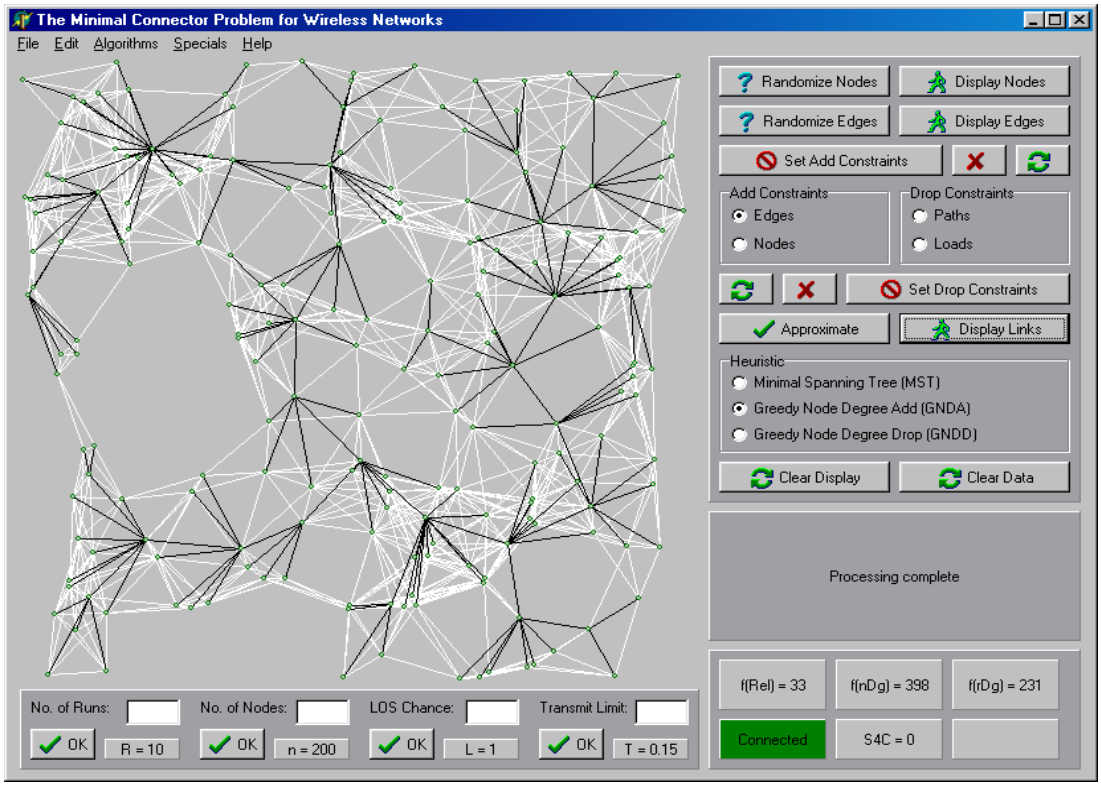



**Figure 8. Add Solution**

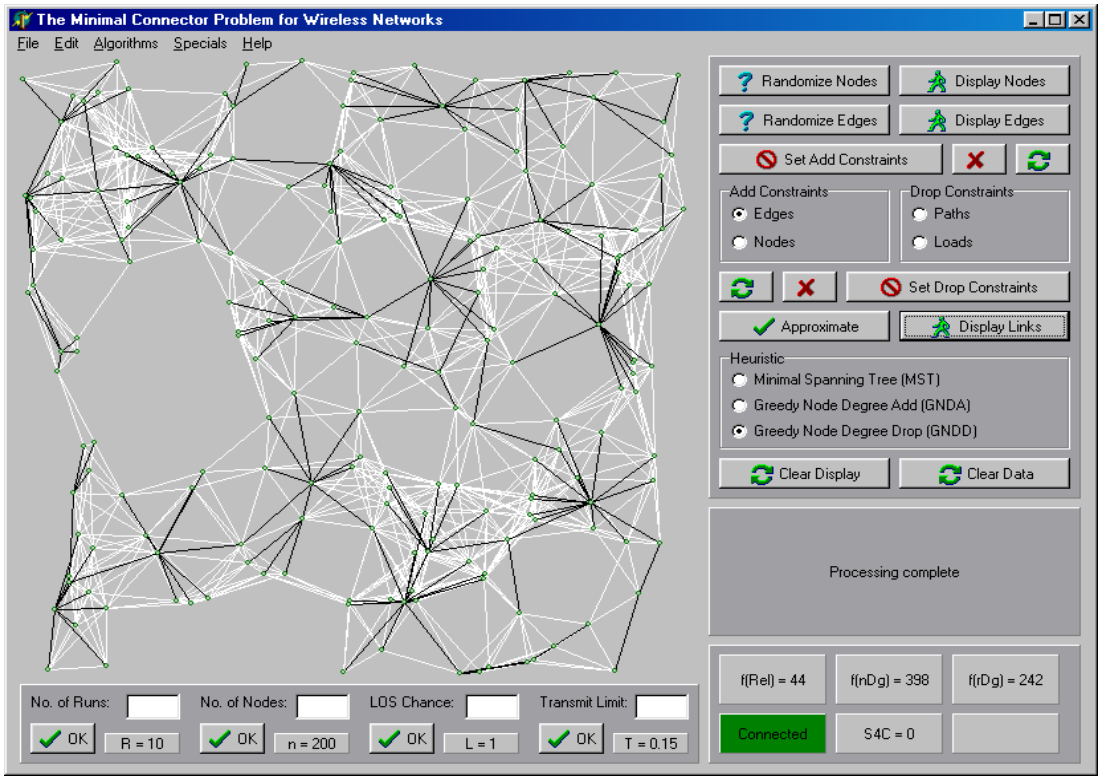



**Figure 9. Drop Solution (1)**

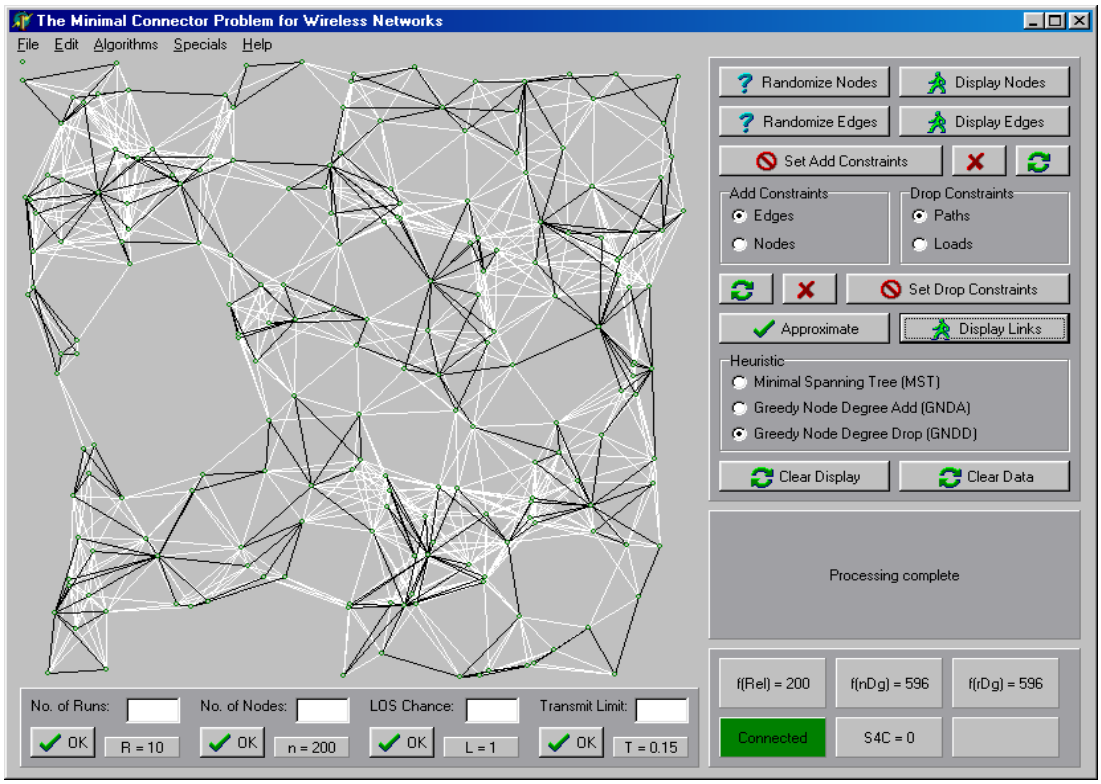



**Figure 10. Drop Solution (2)**

The MST solution is given in Figure 7 with the figures for *f(Rel)* (=$f_{Rel}$), *f(nDg)* (=$f_{nDg}$) and *f(rDg)* (=$f_{rDg}$). Figure 8 shows the equivalent solution produced by the Add algorithm. $f_{Rel}$ has been reduced by approximately four fifths and $f_{rDg}$ by about one third. The value of $f_{nDg}$ remains constant for tree structures. In Figure 9, the Drop algorithm has been applied with the same constraints. It also performs better than the MST but not quite so well as the Add. Figure 10 shows the same node/edge set with the additional constraint of a minimum vertex degree two ($\rho_i = 2$) for all nodes, $i$ (so $f_{Rel} = n$). These examples are broadly typical of the results summarised in Table 1.

Table 1 compares the results of the Add and Drop algorithms for *f(Rel)* (=$f_{Rel}$) and *f(rDg)* (=$f_{rDg}$) with those from the MST. A number of runs were used for each l-o-s and transmit limit combination, as described above. Each figure is shown as a (mean) ratio of the objective number to that obtained with the MST, lower values showing the greater improvement. Each pair compares the mean result for the Add and Drop algorithms. Both algorithms improve as the valid edge set approaches full-feasibility since solution networks may tend toward the star network ideal of Section 2. (For example in Table 1, for $n = 100$ with line-of-sight ($L$) probability *0.5* and transmit limit ($T \equiv d_{max}$) *0.5* (of the unit square), the Add algorithm reduces the number of relays to *12%* of that in the MST solution and the total relay degree to *65%* compared with the MST. The equivalent figures for $L = 1.0$ and $T = 1.0$ are *1%* and *57%*.) However, both algorithms work reasonably well for a variety of constraints. $f_{nDg}$ comparisons are only meaningful for problems constrained with drop constraints and are not included in these figures.

In all cases tested, the Add algorithm produced the best results and is to be preferred to the Drop algorithm where its use is adequate. That is, add constraints permit the use of the Add algorithm while (add and) drop constraints require the Drop algorithm to be used.

## 5.2. Efficiency

This paper is intended to discuss accuracy of results rather than computational complexity. The programs used for algorithmic comparison are simple, written in a high-level language and largely non-optimized. However, in conjunction with theoretical analysis, some guarded observations are possible although much is dependent upon applied constraints, including l-o-s characteristics, transmission limits and node, path and load restrictions. Table 2 summarises run times for both (Add and Drop) algorithms implemented in Delphi Pascal running on a 2.8GHz desktop processor. Shorter run times result from lightly- or un-constrained test cases and longer run times from heavily-constrained test cases.

Prim's algorithm may be implemented in approximately $O(n \log n)$ steps. In principle, the Add algorithm has a similar (worst-case) complexity. However, in practice it appears to function considerably better, due largely to the tendency for many nodes to be added to the spanning tree at each stage. For a graph of *1000* nodes, the Add algorithm ran around six times faster than the MST algorithm (approximately *1* second compared with approximately *6* seconds) and the factor increases for larger values of $n$. The Drop algorithm, in comparison, must test for connectivity and load constraints before removing a link and has (worst-case) complexity of $O(n^3)$. For most practical constraint sets, however, it appears closer to $O(n^2 \log n)$. This is still significantly greater than the Add algorithm, which should be used in all cases where drop constraints do not apply.

**Table 2. Run Times**

Borland Delphi (Ver. 6) Pascal
2.8 GHz Pentium processor (1MB RAM)
(non-optimised code)

| No. of nodes ($n$) | Add Algorithm | | | Drop Algorithm | | |
|---|---|---|---|---|---|---|
| | Best case | Mean | Worst case | Best case | Mean | Worst case |
| 30 | < 1 | < 1 | < 1 | < 1 | < 1 | ≈ 1 |
| 100 | < 1 | < 1 | ≈ 1 | 2 | 3 | 6 |
| 300 | < 1 | ≈ 1 | ≈ 1 | 28 | 35 | 47 |
| 1000 | ≈ 1 | ≈ 1 | ≈ 2 | 412 | 502 | 734 |

Run times (total elapsed program time) in seconds ($s$)

## 6. CONCLUSIONS

The graph-theoretic concepts discussed here are well established. This paper has introduced the necessary variants and constraints for their application to the design of wireless networks. These restrictions require a re-tooling of existing solution methods. The techniques given in this paper give good results but are not necessarily the last word in algorithmic design. It is proposed that further refinement is possible based on a deeper consideration of particular constraint sets. Enhanced constraint formulisations, based on further considerations such as frequency/bandwidth limitations are also proposed.

## 7. ACKNOWLEDGEMENTS

The comments and suggestions offered by the referees have been invaluable in preparing the paper for publication and were gratefully received. Thanks also to Mike Headon for proof reading the final version.

## 8. REFERENCES

Baldi, P., De Nardis, L. & De Benedetto, M-G. (2002) *Modeling and Optimisation of UWB Communication Networks through a Flexible Cost Function*, IEEE Journal on Selected Areas in Communications, Vol. 20, No. 9, December 2002, pp1733-1744.

Dijkstra, E.W. (1959) *A Note on Two Problems in Connexion with Graphs*, Numerische Mathematik, Vol. 1, pp269-271.

Du, D-Z. & Pardalos, P.M. (1993) *Network Optimization Problems: Algorithms, Applications and Complexity*, World Scientific.

Fowler, T. (2001) *Mesh Networks for Broadband Access*, IEE Review, January 2001, pp17-22.

Garey, M.R. & Johnson, D.S. (1979) *Computers and Intractibility: A Guide to the Theory of NP-Completeness*, Freeman.

Grout, V.M. (1988) *Optimisation Techniques for Telecommunication Networks*, Ph.D. Thesis, Plymouth Polytechnic UK.

Harary, F. & Palmer, E.F. (1973) *Graphical Enumeration*, Academic Press.

Kershenbaum, A. (1993) *Telecommunication Network Design Algorithms*, McGraw-Hill.

Kershenbaum, A. & Chou, W. (1974) *A Unified Algorithm for Designing Multidrop Teleprocessing Networks*, IEEE Transactions on Communications, Vol. COM-22, No. 11, pp1762-1772.

Luo, X., Yan, P., Guo, C. & Tang, Y. (2002) *Optimal Placement and Deployment Strategies in Mobile Agent-Based Network Management*, IEEE International Conference on Communications, Circuits and Systems and West Sino Expositions, Vol. 1, 29th June-1st July 2002, pp753-757.

Moon, J.W. (1970) *Counting Labelled Trees*, Canadian Mathematics Congress, Montreal.

Papadimitriou, C.H. & Steiglitz, K. (1998) *Combinatorial Optimzation: Algorithms and Complexity*, Dover.

Prim, R.C. (1957) *Shortest Connection Networks and Some Generalizations*, Bell Systems Technical Journal, Vol. 36, pp1389-1401.