ELSEVIER

Computer Communications 20 (1997) 521–527

computer communications

Fast heuristics for practical TDM network design

G.P. Ray[a], P.W. Sanders[b,*], C.T. Stockel[b]

[a]AT&T, Redditch, UK
[b]University of Plymouth, Drake Circus, Plymouth PL4 8AA, UK

Received 2 November 1993; revised 14 April 1995; accepted 14 April 1995

Abstract

Many large scale core data networks are now based on the new generation of intelligent time division multiplexers. These allow permanent virtual circuits between any network nodes and automatic rerouting of lost connections should link failures occur. The design requirements for such networks are very different to those more usually associated with standard packet switched network arrangements. For a given traffic matrix the number of possible network topologies, link capacity selections, and circuit routings, even between a small number of nodes, is relatively large. When taking into account the additional requirements of optimising the overall link capacity of a network and ensuring its resilience to failure by specifying link disjoint primary and backup paths, the design complexity becomes very large. It is also necessary to use commercial link costs for all the link specifications likely to be selected from the various PTT service providers to ensure realistic design results.

The following paper describes the modules of a heuristic method that has been successfully used on a standard personal computer to provide minimum cost optimum designs based on the described network characteristics. The heuristics incorporate a speed versus accuracy trade-off factor so that rapid approximate designs may be examined for differing traffic conditions. © 1997 Elsevier Science B.V.

Keywords: Network design; Optimisation; Link capacity

1. Introduction

Many large scale networks are now being provided incorporating intelligent bandwidth management systems based on time division multiplexing (tdm). A tdm link supports a number of channels between nodes and a concatenation of channels allows a connection over a multinode path. These channels might support a range of data and voice services. The intelligence of these systems is based on the operation of control software at each node whereby minimum 'cost' paths are selected, and the rerouting of channels is performed automatically in the event of a link failure. This can only be achieved if sufficient capacity is available on alternate paths, and it is usual for priority systems to be implemented to determine which channels are lost if there is insufficient capacity. These systems differ from the dynamic capacity allocation of packet switched systems in that they use dedicated capacity for each channel. In a packet based system it is the packet transit time, across the network, that degrades as capacity becomes limited following link failures, however, in these tdm systems a blocking arrangement occurs with some calls being lost while others maintain full service with the same delay.

This paper considers the design of large scale tdm networks, using a method developed for a standard PC yet still able to cope with designs of 20 or more nodes. This compares very favourably with other network design techniques [1]. In our design methodology the traffic requirements are measured in terms of 64 kbps channels since this was the smallest channel size and all other channels are multiples of 64 kbps, though it is possible to use any discrete capacity values. To date the work has been applied to the design of European digital networks with link capacities up to 2 Mbps.

In designing any computer network there are a number of trade-offs that can be made in order to meet the objective of a minimum overall cost. In selecting the optimal set of links from a myriad of possible choices, a rapid means is useful for identifying those which are likely to be optimal. Merely selecting minimum cost links until a 'sufficiently' connected topology is reached, or all traffic demands can be routed, cannot be considered a very efficient design method, because of time constraints. The definition of 'sufficient connectivity' is usually based upon a reliability factor required of the network. Typically, in order to meet a

* Corresponding author. Tel: +44 1752 232572; fax: +44 1752 232583.

0140-3664/97/$17.00
PII S0140-3664(97)00046-7

required service availability it is necessary to provide a minimum of two links connected to each node. The work described in this article assumes a minimum of two link disjoint paths are needed for all virtual circuits that use the individual channels. A primary path meets the traffic demand for source–destination end points of the traffic matrix under normal conditions, but when a fault occurs on any link or at any node along this path, a secondary path is utilised to meet the service.

It has been recognised that the network design problem can be very difficult; Johnson et al. [2] showed that it may belong to the NP-hard class of problems, the time to solution for such problems is not a polynomial function of the problem size. The solution time increases exponentially with increasing problem size, which involves the number of nodes, range link capacities, a number of special requirements etc. For this reason heuristic techniques have been investigated for a practical solution.

Many attempts have been made to design both packet switched and tdm networks using linear programming (LP) methods to create initial feasible topologies, followed by perturbation methods to reduce the level of redundant capacity [3,4]. The function of the linear program is to create an initial feasible topology of low cost and it is the quality of the perturbation scheme that one depends upon to drive the topology cost down to the optimal solution. The problem is that so many network topologies can be created with an initial 'low cost' and any attempt to drive one in particular to an optimum may find a 'local minima' in the sample space. The perturbation scheme must then be capable of generating a perturbation sufficiently large to escape any local minima without losing any 'optimal components' of the original design.

An alternative system to linear programming for finding initial feasible topologies is the threaded search technique, which describes a repeated application of a search for the next best path to allocate at each stage, as the initial design is produced. The problem's difficulty then becomes one of determining what constitutes the basis for selecting the best path, and then a method for dealing with the large number of possibilities that can result. Even though the sample space for the general network design is very large, it is possible to develop a number of heuristic methods to select the topology features that are, from experience and analysis, known to be poor. Such a system, being based upon step-by-step link selection decisions, will suffer from a 'lack of foresight', making short term opportunist selections without regard to the global optimum solution. Using a method that bases the primary selection of paths on minimum cost (a greedy algorithm) does not necessarily lead to an ultimate minimum cost solution [5]. The use of lookahead techniques, which examine the possibilities of all outcomes and their consequences, for a number of selections in advance, is well established. However, these can suffer from an expansion of the complexity and usually cannot be performed directly by brute force.

2. The threaded search structure

Feasible topologies may be produced by the systematic search and allocation of all paths, evaluating their apparent cost, and then selecting the cheapest. The advantage of this method is that capacity is only sought for the active paths that meet the traffic matrix requirements and the traffic is allocated capacity on the links in a sequence determined by successively determining the cheapest path. The 'thread' of the design process is the order in which the selections are made. It is continuous for the initial topology design phase, since no allocated path undergoes perturbation until all traffic demands have been assigned a path and the necessary link capacities. In order to meet the requirement for resilience to link failure on the primary path, it is necessary to include the allocation of capacity for the link disjoint secondary path in the threaded search. In making the repeated selection of the next cheapest path it is possible to simultaneously include the search for the secondary paths and allocate the cheapest whether they are found to be primary or backup. Since this is essentially still a 'greedy method' it is then necessary to apply a heuristic technique to counter the lack of any lookahead. The path selection process is now examined in more detail.

3. Selecting routes

In order to see how a topology is created by a threaded search for best primary or secondary paths it is necessary to examine the way in which the cost function influences their selection.

The unit channel cost c between nodes i and j is given by;

$$c_{ij} = l_{ij}[k_0(\Omega_{ij}) + d_{ij}\Omega_{ij}f(\Omega_{ij})]$$

where $\Omega_{ij} = \alpha_{ij} + \beta_{ij}$

- d_{ij} is the distance from node i to node j;
- l_{ij} is a [0,1] matrix representing the presence of link i–j;
- Ω_{ij} is the allocated traffic variable representing the sum of primary and backup allocated channels, in 64 kbps units;
- α_{ij} represents the primary traffic on link ij;
- β_{ij} represents the secondary traffic on link ij;
- k_0 is the fixed link setup cost for the channel capacity W_{ij};
- $f(\Omega_{ij})$ is the cost rate of change function. It determines the change in per channel cost with increasing channel capacity.

To define the cost of a path, let vector $\boldsymbol{R}$ be a series of L nodes between which the links form a single path such that

$$\boldsymbol{R} = \{r_1, r_2, r_3, \ldots, r_L\}$$

where links in the path are given by r_i and i denotes the ith link from node $[i]$ to $[i+1]$.

The absolute cost of a path is the sum of all included link costs, from r_1 to r_L. This assumes that no other connections are currently made in the network.

$$c(\mathbf{R}) = \sum_{i=1}^{L-1} [k_0(\Omega_{r_i r_{i+1}}) + d_{r_i r_{i+1}} \Omega_{r_i r_{i+1}} f(\Omega_{r_i r_{i+1}})]$$

The path cost is dependent upon the total number of links involved and the allocated capacity of each link making up the path. This capacity consists of two parts; the already allocated capacity prior to the new path being added and that of the path itself, which is a constant value at the time of comparison between alternative paths for a chosen traffic demand.

In Europe, the international link tariffs show a predominant drop in unit channel costs with increasing capacity. This leads to a downward force on costs as the allocated capacity Ω of a link increases. Allocating more capacity to links, within limits, can therefore reduce overall channel costs at future stages of the design process. However, the selection of minimum cost paths at each stage is desirable as long as they contribute to overall optimality.

Where a small number of paths fall within a narrow cost bound the cost differential may be insufficient to justify selecting merely the very lowest cost without using a look-ahead method. Under these circumstances an additional, or secondary criteria for selection may be employed. Returning to the path cost equations above, it can be seen that this second criteria involves increasing the capacity allocation on some links. Given a number of similarly priced possible paths for a given connection the one that increases the capacity allocated to links across the network may be the most advantageous.

The path selection process therefore consists of a search for all feasible paths and a repeated comparison that tests for those within a cost bound. This bound is referred to as the 'cost desensitivity' because of its effect in reducing the path selection algorithm's sensitivity to minimum cost paths. The cost comparison is therefore based on the following test;

$$\left| \sum_{i=1}^{L_1} (k_0^{L_1} + d^i \Omega^i f(\Omega^i) + - \sum_{j=1}^{L_2} (K_0^{L_2} + d^j \Omega^j f(\Omega^j) \right| \le \Delta$$

where Δ is the desensitivity factor.

The setting of this desensitivity factor to produce the optimum network has been found to be dependent upon the particular traffic values and link cost conditions under investigation, but the requirement that the optimal setting be found proves very useful in network design problems. The desensitivity parameter gives the network designer a 'tuneable' control with which to generate slightly differing network topologies without needing to adjust the input data, i.e. the traffic matrix and link tariffs. The desensitivity factor controls the way in which additional capacity is encouraged in the topology at each step of the design. The typical link capacities available across national boundaries in Europe are 64 kbps, 128 kbps, 256 kbps, 384 kbps, 512 kbps, 768 kbps, 1024 kbps, 1536 kbps and 1984 kbps, or

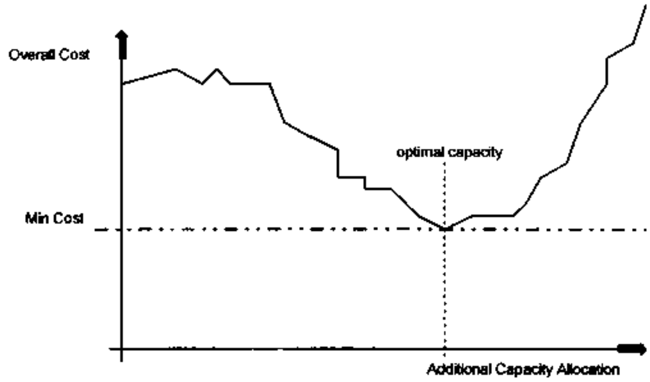

Fig. 1. Increasing total capacity allocation first improves, then degrades overall topology cost.

expressed as multiples of 64 kbps, 1, 2, 4, 6, 8, 12, 16 and 31. In the process of a network design cycle where new links are to be added to support a given capacity it is frequently likely that a capacity will be required that is not directly available from the PTTs, the next biggest available capacity will instead be selected. This leads to the inclusion of spare capacity in the design. From the perspective of overall topology cost, there is a balance to be found towards the optimal level of additional capacity to be allocated, since unallocated capacity added early in the design cycle may well be beneficially used in later path selections, but the likelihood of it being used decreases as the design nears completion.

The total network cost at any given stage of the optimisation, step n, can be written by replacing all static variables with the dynamic (n) time indices;

$$\psi(n) \sum_{i=2}^{N} \sum_{j=1}^{i=1} l_{ij}(n), [k_o(\Omega_{ij}) + d_{ij} \Omega_{ij}(n) f(\Omega_{ij}(n))]$$

where $\Omega_{ij}(n) = [A_{ij}(n) + B_{ij}(n)]$.

$$\psi(n) = K_0 + \sum_{i=2}^{N} \sum_{j=1}^{i-1} d_{ij} \Omega_{ij}(n) f(\Omega_{ij}(n))$$

where the $l_{ij}(n)$ is a 0,1 matrix to represent whether a link from node i to node j is in place at stage n of the design process.

Fig. 1 indicates how, when the overall topology cost benefits from the lowered unit channel costs, the cost desensitivity can encourage the allocation of extra capacity. This effect is only positive when the cost penalty is small. If the desensitivity to cost is made too large then excessive capacity can be incurred that contributes to an increasing overall topology cost and does not produce the optimum topology.

The manner in which this deliberate over-capacity in the early stages of the topology creation can influence the topology is illustrated in Fig. 2.

To simplify the analysis consider the link AB to have zero capacity allocated initially. If the capacity allocation on links AC and BC is greater than AB then assume that their unit channel cost is less than that of AB, per unit

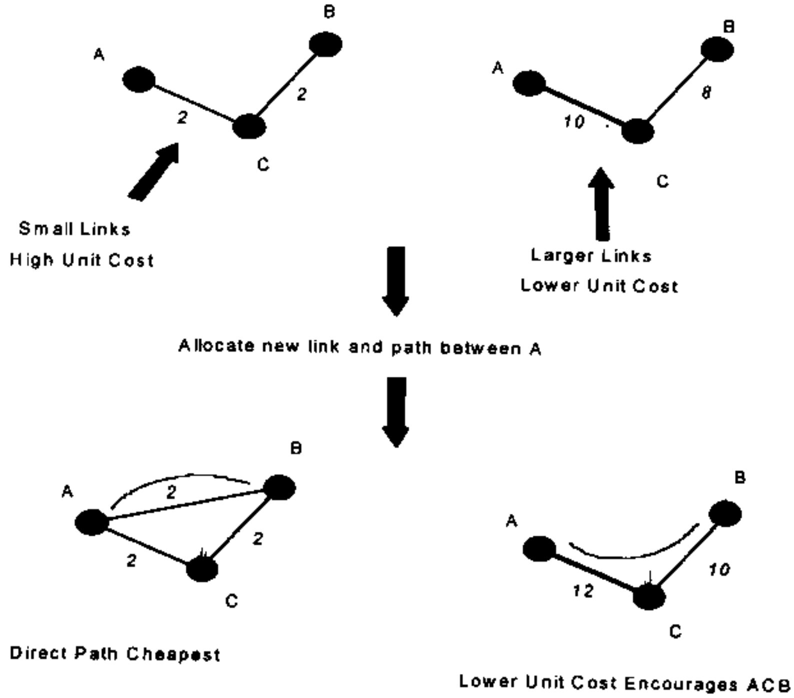

Fig. 2. Trunks of greater capacity have lower channel costs and can favour paths of greater length.

distance. The increase in distance that the path ACB may incur is dependent upon this difference in unit channel cost. Define the change in link i–j cost as $\delta_{ij}(D)$ where D is the change in allocated channels.

Thus

$$\delta_{ij}(D) = k_0(\Omega_{ij} + D) + d_{ij}(\Omega_{ij} + D)f(\Omega_{ij} + D) - [k_0(\Omega_{ij}) + d_{ij}\Omega_{ij}f(\Omega_{ij})]$$

In order that the bound on the excess distance via ACB be found, the cost of paths AB and ACB are equated, i.e. when the cost is the same, the limit of the boundary is defined. An expression can therefore be written where the boundary is defined as the point at which the cost increment for paths allocated A–B equals that for A–C–B:

$$\delta_{AB}(D) = \delta_{AC}(D) + \delta_{CB}(D)$$

which may be expanded to

$$k_0(\Omega_{AB} + D) - k_0(\Omega_{AB}) + d_{AB}[(\Omega_{AB} + D)f(\Omega_{AB} + D) - \Omega_{AB}f(\Omega_{AB})] = k_0(\Omega_{AC} + D) - k_0(\Omega_{AC}) + d_{AC}[(\Omega_{AC} + D)f(\Omega_{AC} + D) - \Omega_{AC}f(\Omega_{AC})] + k_0(\Omega_{CB} + D) - k_0(\Omega_{CB}) + d_{CB}[(\Omega_{CB} + D)f(\Omega_{CB} + D) - \Omega_{CB}f(\Omega_{CB})]$$

Rearrangement produces the general equation describing a constant K of the form:

$$K = d_{ij}vf(v) + d_{jk}wf(w)$$

The right hand side now gives rise to a series of contours contingent upon d_{ij} and d_{jk} and the v and w values.

Fig. 3 shows the range of possible positions node j may take, for each capacity C_1 to C_5, with respect to I and k, when considering the region for alternate paths. The boundary represents the cost contour of each capacity. It is possible to examine any given network and evaluate the possible cost trade-offs of all possible paths, and hence the potential saving, by selecting paths with the greatest number of links. Increased capacity and attendant reductions in unit channel cost are utilised to produce the overall cost savings.

This may be extended to the variable case where the links AC, CB and AB may take on different capacity values and here the boundary of the ellipse will change. Between any pair of nodes with a given link capacity it is possible to calculate the maximum extra distance beyond the direct path length. This takes into account the unit channel cost

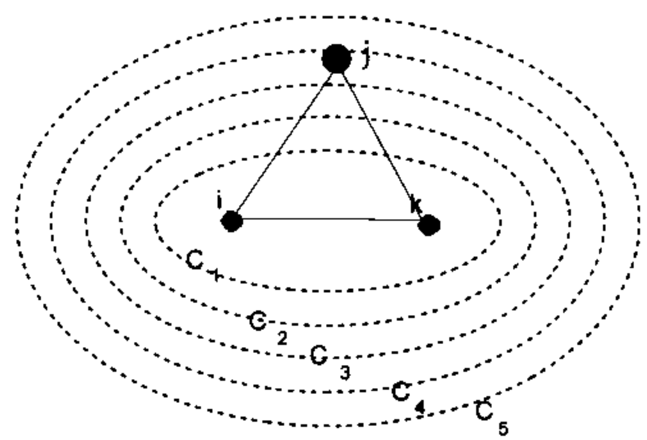

Fig. 3. Ellipses of constant inter-nodal distance.

reductions gained by sharing other nearby links that are available, and gives rise to the regions within which intermediate nodes may be used for routing. For a simple single intermediate node, assuming equal channel costs for the two link alternative path, the following applies:

$$X_{\cdot c_1} \leq Y_{\cdot c_2}$$

where

c_1 = unit channel cost of the direct path;
c_2 = unit channel cost of links between intermediate nodes;
X = distance between nodes A and C;
Y = total distance between A, B and C.

In order to reduce the number of paths to be searched it is possible to limit the examined nodes for possible intermediate hops. Reducing the path-finder complexity a fast method for determining a reduced set of nodes. A net gain in speed must not be lost in the 'speed up ' algorithm itself, of course.

Empirical results show that the most likely regions in which paths are formed between two nodes mark an approximate ellipse around the end nodes. It is possible to very quickly determine the 'ellipse bound nodes' for the path-finder by using the following test:

For end nodes A and B separated by a distance d_{AB} consider a candidate node X, having distance d_{AX} and d_{XB} from nodes A and B respectively. The ellipse is defined by a factor (of the AB node separation. i.e. node X is within an ellipse ρd_{AB} if

$$\rho d_{AB} \geq d_{AX} + d_{BX}$$

In practical design situations for international networks, link tariff structures bear little resemblance to physical distance and it is possible to replace all distance variables with those of corresponding cost. The 'elliptic bound' when used in path searches has been found to rapidly identify good candidate intermediate nodes and has an increasing benefit as the number of network nodes increases (Figs 4 and 5). The complexity of the path searching algorithm, when examining all network nodes, N, is a function of the number of permutations of M nodes from N, where M is the maximum number of intermediate nodes. If however the 'elliptic bounding' is used to select E candidate nodes the number of permutations is a function of the number of combinations of M from E, where $M \leq E \leq N$.

4. Perturbation schemes

Owing to the threaded path selection process and discrete link capacity availability it is possible that network capacity is allocated in a less than optimal manner but any excess may be identified using perturbation schemes. These aim to select high cost (or low efficiency) links and replace them with cheaper (or more efficient) alternatives. The means for identifying single or groups of links for deletion is usually made on the basis of link utilisation. Some approaches such as the branch exchange or the cut saturation algorithms [4] look for minimum efficiency links to delete and the minimum cost alternatives to restore the traffic loss.

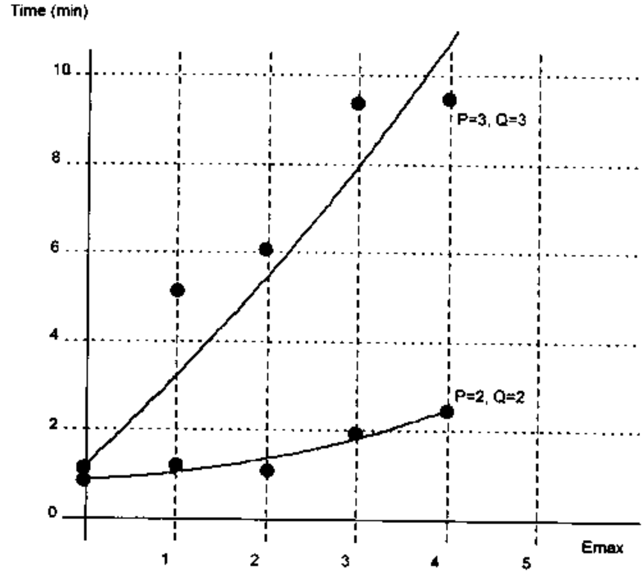

Fig. 4. Graph to show the design times for 8 node topology.

Two perturbation schemes have been found to be particularly effective for use with the threaded design method. The first is based on identifying any link with less than 75% utilisation. A list is built up of such links and they are deleted in turn, the displaced traffic is then re-routed according to the minimum cost path available. It is often found that the original link is restored, but savings are consistently found for a large number of topologies. The second perturbation scheme identifies any link to which a single path is allocated. The link is deleted and the minimum cost alternative path is sought. Very often it has been found that sufficient spare capacity exists within the topology to accommodate the paths displaced by the link deletion.

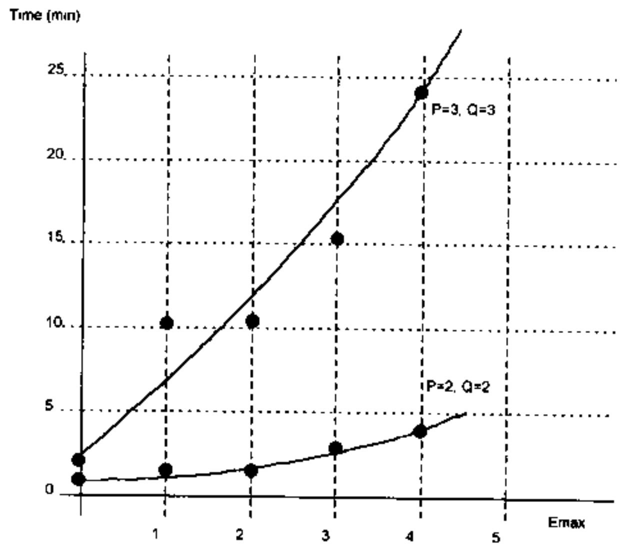

Fig. 5. Graph to show the design times for 9 node topology.

Table 1
$N = 8$, cost and design time

Nodes	Hops	E_{max}	Cost (£)	Design time (s)
8	2	0	116,114	41
		1	100,219	94
		2	108,114	84
		3	108,114	109
		4	108,114	132
	3	0	100,219	93
		1	102,577	339
		2	101,382	370
		3	91,212	565
		4	91,212	565

Table 2
$N = 9$, cost and design time

Nodes	Hops	E_{max}	Cost (£)	Design time (s)
9	2	0	139,834	63
		1	132,294	146
		2	134,663	131
		3	134,663	184
		4	134,663	233
	3	0	132,294	146
		1	129,570	612
		2	127,388	640
		3	116,923	1012
		4	116,923	1437

The problem with each of these schemes is that, where more than one candidate link is found for deletion there is the problem of selecting which order to perform the delete and add. It is necessary to repeat the perturbations until no further improvement is found. The ordering problem can mean this extends the run time.

5. Results

The speed advantage of the elliptic bound is illustrated below. Tests were performed on a low powered, 386DX IBM-PC. The following variables are used:

P = maximum number of hops allowed in a primary path;
Q = maximum number of hops allowed in a backup path;
E_{max} = elliptic bound, number of candidate nodes for path search in addition to P or B for primary or backup paths respectively.

The paths are searched in order, starting with 1 hop paths increasing up to P_{max} maximum primary hops and Q_{max} maximum backup hops. The elliptic bound is described by the value E_{max} which represents the maximum number of nodes between which to search for paths, in addition to those allowed for by the P_{max} and Q_{max} parameters. The number of nodes used in the path search is therefore a maximum of $P_{max} + E_{max}$:

e.g. if $P_{max} = 3$ and $E_{max} = 2$, then the elliptic bound is extended to encompass the 5 nearest nodes to the end points falling within the ellipse.

In terms of computational complexity this therefore implies that the work required to search for paths with a maximum of 3 hops, where $E_{max} = 2$, is equivalent to searching for 5 hop paths with $E_{max} = 0$.

The cost improvements as a result of expanding the maximum number of hops per path is illustrated in Tables 1 and 2. The design times indicated in Figs 4 and 5 reflect the total time taken to create the initial topology and then perform perturbations until no further topology cost improvement was obtained.

The effects of the desensitivity factor on network cost are shown in Table 3. While there is no direct relationship between the desensitivity and minimum network topology cost, there is a definite range of values for each topology that yields improved designs. Additional columns in the table are added to indicate the cost of the capacity allocated but unused, the percentage cost utilisation and the percentage capacity utilisation. The last column shows that the ratio of capacity allocated to the primary path compared with that for the backup path.

Table 3
A range of 14 node network solutions

Design Topology	Desens (%)	Cost (£)	Spare Value (£)	Cost Util (%)	Capacity Util (%)	P/Q
T-1	0	151,476	30,969	79.6	76.6	1.57
T-2	10	145,417	33,834	76.7	72.6	1.11
T-3	20	151,083	22,270	85.3	83.0	1.11
T-4	30	140,556	22,153	84.2	81.1	0.90
T-5	40	144,553	18,166	87.4	83.9	0.74
T-6	0	146,291	21,116	85.6	83.8	1.38
T-7	10	145,873	30,347	79.2	75.2	1.31
T-8	20	141,002	23,848	83.1	80.0	1.37
T-9	30	139,284	20,480	85.3	80.2	1.20
T-10	40	141,711	22,428	84.2	81.0	1.08

Util = utilisation; Desens = desensitivity factor.

6. Conclusion

The core network design problem for tdm systems can achieved using a new effective method. Although using heuristic techniques the consistency of the designs gives an indication that the methods are soundly based. Good network designs have been generated, to a quality suitable for use by a major international network provider. The method offers an alternative solution to a problem previously only tackled using systems based upon linear programming and requiring considerable mainframe or workstation processing power. Two new techniques have been developed that allow similar sized problems to be tackled on medium power, personal computers. The first technique, called 'cost desensitivity', improves the quality of designs using greedy algorithms to sacrifice short term gains for a global topology objective. The second technique used in path selection and called 'elliptic bounding', greatly reduces the execution time of searches for feasible paths by forming a logical limit to the number of nodes between which paths are tested. Using the simple equation of the ellipse, the test itself adds an insignificant overhead to the overall calculation. The speed improvements of the elliptic bound are related to the size of the boundaries; the greater the speed the less likely are the paths to be optimal. However, network designs can be constructed rapidly using these fast search methods that provide cost estimates to within typically 10% of a solution's final cost using a slower, more comprehensive search.

This type of design tool is entirely acceptable in practice since the designer is given full control over the network parameters and also the opportunity to make any manual adjustments. Although optimality is not guaranteed different network designs may be compared quantitatively with detailed figures showing the breakdown of both financial and capacity utilisation for all the network traffic.

Acknowledgements

The authors wish to thank John Allen of AT&T BCS-E for his advice and suggestions in the preparation of this manuscript.

References

[1] Y.K. Agarwal, An algorithm for designing survivable networks, AT&T Tech. J., 68(3) (May-June 1989) 64–76.
[2] Johnson, Lenstra, Rinooy Kan, The complexity of the network design problem, Networks 8 (1978) 279–85.
[3] H. Frank, W. Chou, Topological optimisation of computer networks, Proc. IEEE 60 (1972) 1385–1397.
[4] M. Gerla, L. Kleinrock, On the topological design of distributed computer networks, IEEE Trans. Commun. COM-25 (1) (1977) 48–60.
[5] B.M.E. Moret, H.D. Shapiro, Algorithms from P to NP, vol. 1, Benjamin/Cummings, Redwood City, 1991.