OPTIMISATION TECHNIQUES FOR TELECOMMUNICATION NETWORKS

V.M. Grout

March 1988

Department of Communication Engineering
Faculty of Technology
Plymouth Polytechnic

in collaboration with

British Telecom Tallis Consultancy
London

This thesis is submitted to the Council for National Academic Awards in partial fulfilment of the requirements for the degree of Doctor of Philosophy

DECLARATION

I hereby declare that while registered as a candidate for the degree of Doctor of Philosophy with the Council for National Academic Awards, I have not been a registered candidate for another award of the Council or of any other academic or professional institution.

[illegible]

Vic Grout

March 1988

To Mum and Dad

Optimisation Techniques for Telecommunication Networks

V.M. Grout

Abstract

This thesis deals with various facets of the optimisation problem for telecommunication networks and proposes a number of new techniques for their solution.

The necessary essentials, Graph Theory, Complexity Theory and Telecommunication Principles, are investigated. The relevant graphs are enumerated and the requirements of suitable optimisation algorithms for certain graphical problems are established. The Private Automatic Branch Exchange (PABX) is introduced. The variety of telecommunications traffic as well as the practical requirements of a connection topology are discussed.

The fundamental Network Optimisation Problem (NOP) is defined and analysed. Simple exhaustive methods of solution are considered together with partial solution algorithms and simplification methods. Centralised networks with and without concentrators are introduced. Extensions and modifications are proposed for some techniques and existing practical methods of dealing with the NOP are investigated.

A number of new ideas are proposed for the practical solution of the NOP. Reduction methods are presented for replacing large unmanageable networks with smaller ones, on which optimisation can take place. Fixed topology techniques are introduced for initial tandem switch selection purposes and perturbation methods are considered which can be applied to such an initial solution. Lookahead methods of link removal are introduced for the purposes of determining the tandem interconnection network together with the traffic routeing strategy. A composite method is proposed incorporating all of these concepts and the results of a number of numerical experiments upon actual network problems are presented.

The extension of the proposed techniques to other areas of problem solving and optimisation is considered. In particular, a new method for the solution of the Euclidean Travelling Salesman Problem (ETSP) is presented.

A brief discussion is undertaken, in conclusion, concerning the practical difficulties of the NOP and the restrictions thus placed upon solution algorithms of various types.

Preface

The work contained in this thesis constitutes the results of a three year investigation into various aspects of the telecommunication network design problem, undertaken within the Department of Communication Engineering, Plymouth Polytechnic, UK, under the supervision of P.W. Sanders (Communication Eng.) and C.T. Stockel (Computing). The project began on September 24th, 1984.

The work in the first two years was carried out under the sponsorship of Tallis Consultancy, London, UK, the network consultancy service for British Telecom. In return for this sponsorship, a software package was produced, enabling network design, which is now in use by Tallis both internally and externally (Grout [1987a & 1987b]).

With this part of the project completed, a Local Education Authority grant was obtained for the final year.

Two computers have been used, on the whole, throughout this time: the multiple-processor PRIME 9950/9755/850/9750/250 of the Polytechnic's Computing Service (Prime [1981]) and an IBM PC-XT (IBM [1983]). The language used in all experiments was Pascal (Findlay & Watt [1978]).

This thesis has been prepared using the DW2 word processing package and the Lettrix print enhancement system upon the IBM PC-XT.

Acknowledgements

To embark upon a lengthy course of work in which, by definition, only a limited knowledge of future developments is available at any stage is large step to take in itself. To expect to achieve anything in this context without assistance from others would be worse than naive. Help and advice from those willing to give it is not only most welcome - it is essential. I am glad to say that I have been entirely fortunate in this respect.

Firstly, sincerest thanks are due to my supervisors Peter Sanders and Colin Stockel. There would be have been no project undertaken without Peter's enthusiasm and I have always greatly appreciated his selfless commitment to the work in hand. Colin, meanwhile, has been a constant source of sensible advice in all situations.

The assistance provided by the collaborating establishment, Tallis Consultancy, both financial and technical, was obviously a major factor in the success of the project undertaken. In particular, I would like to thank Ian Upton, Paul Reynolds and David White for their friendly involvement.

I am grateful for the superb cooperation I received from the technicians within the Department. Many hardware problems were overcome through their professionalism, particularly that of Alan Santillo and Steve Roberts. The library and computing staff within the Polytechnic have also been of consistent help.

The greatest thanks of all, however, must be to my family for giving me the help, love and warmth which kept me going during the less tranquil times, both within the last three years and before. I will never forget it.

CONTENTS

TABLE OF FIGURES

TABLE OF MAJOR SYMBOLS AND TERMS

A	:	The area covered by a network
AAP	:	Asymmetric assignment problem
ATSP	:	Asymmetric travelling salesman problem
b(x, y)	:	Probability that a node a distance y from a node a distance x from the centre of its group is not in the same group
C(N)	:	The cost of a network N
C_n	:	The number of connected labelled graphs on n nodes
DOR	:	Degree of rerouteing: the maximum number of extra tandems in the path between two tandems
E	:	The expected number of errors during the reduction process
ETSP	:	Euclidean travelling salesman problem
f(x)	:	PDF of the distance of a node from the centre of its group
$g_{n'}(y)$	:	PDF of the distance between the two nearest nodes at stage n'
G_n	:	The number of labelled graphs on n nodes
GOS	:	Grade of service
HN_{nm}	:	The number of labelled hierarchical m nets on n nodes
HS_{nm}	:	The number of labelled hierarchical m stars on n nodes
I	:	Input parameter set for an optimisation problem
$\mathcal{I}$	:	Input parameter set for the NOP
$\mathcal{I}'$	:	Input parameter set for the tandem selection of the NOP
ISDN	:	Integrated services digital network
K_n	:	The complete graph on n nodes
LOC	:	Level of connectedness: The minimum degree of each node
M	:	The number of tandems
n	:	The number of nodes in a graph or network
n'	:	The number of nodes in the reduced network under

optimisation techniques to be used (such as assuming constant link costs and using the EMST). Others artificially restrict the number of possible solutions so as to decrease search time. The most effective techniques make no such assumptions, but instead apply a heuristic procedure which approximates the optimum solution to the real problem in an acceptable amount of time. Such methods, however, are not in abundance and those that do exist are generally in operation on machines with extremely large processing capabilities.

The purpose of this thesis, subsequent to discussing and attempting to extend the existing methods, is to propose a number of new, but still realistic, techniques for telecommunication network design. It is one of the major objectives of these methods that, in addition to providing acceptable levels of accuracy, the software used to implement them should be capable of running within the confined storage and processing environment of a small machine - a personal desktop computer for example.

1.1. Layout of the Thesis

The search for algorithms which solve various components of the network optimisation problem (NOP) requires a working knowledge of three distinct areas: the theory of algorithms, the principles of telecommunication networks and the underlying concepts behind networks - graph theory.

Chapter 2 discusses the important elements of graph theory, including the necessary definitions and some basic graph types. The more relevant graphs are enumerated and certain graphical optimisation problems are investigated.

Problem solving algorithms may differ considerably in detail. The important aspects which decide whether a particular algorithm is acceptable are discussed in chapter 3. This is concerned with the study of the classes P, NP and NP-complete.

Chapter 4 introduces the practical details of telecommunication networks and discusses some elementary concepts such as equipment costs, traffic principles, grades of service, etc.. The private automatic branch exchange (PABX) is then introduced. Its purpose is

		consideration
N_n	:	The number of acceptable solution networks on n nodes
NOP	:	Network optimisation problem
NP	:	Non-deterministic polynomial solvable problem set
NP-complete	:	Set of non-deterministic polynomial solvable problems proven to be equivalent
O	:	Output parameter set for an optimisation problem
$\mathbf{O}$	:	Output parameter set for the NOP
O(f(n))	:	The complexity function
$p_{n'}(x)$	:	Probability, at stage n', that the nearest neighbour of a node a distance x from the centre of its group is outside of the group
P	:	Deterministic polynomial solvable problem set or the probability that at least one node is in error in the reduction stage
P(n')	:	Probability of an error at stage n' of the reduction process
PABX	:	Private automatic branch exchange
q	:	The number of replacement nodes in the reduced network
q_M	:	The number of replacement nodes to be used with M tandems
r	:	The mean group radius or the degree of lookahead
S_n	:	The number of labelled stars on n nodes
TSP	:	Travelling salesman problem
T_n	:	The number of labelled trees on n nodes
w_i, w_{ij}	:	A general weight
x	:	Number of tandems moved in perturbation method
x_i	:	x-coordinate of node i
y	:	Number of tandems connected in perturbation method
y_i	:	y-coordinate of node j
$y_{n'}$	:	The expected distance between nearest neighbours at stage n'
z	:	Number of tandems disconnected in the perturbation method
$\sigma(i)$	:	The degree of the node i
Ω	:	The optimum number of tandems

CHAPTER 1

INTRODUCTION

'Where shall I begin, please your Majesty?' he asked.
'Begin at the beginning,' the King said gravely, 'and go on till you come to the end: then stop'

Lewis Carroll
Alice's Adventures in Wonderland, 1865

Shamos [1978], in discussing various applications of the Euclidean Minimal Spanning Tree (EMST) problem, in his thesis on Computational Geometry, observes, in passing, that

> *"The EMST problem is a common component in applications involving networks. If one desires to set up a communications system among N nodes requiring interconnection cables, using the EMST will result in a network of minimal cost."*

If only the problem were that simple! Unfortunately the practical details involved in minimising network costs are far more difficult. The cost of linking together two nodes is generally a complex calculation based upon the traffic flowing between them as well as the distance. This cost is rarely linear in distance nor can it be calculated prior to an optimisation algorithm since the traffic the link carries will depend upon the topology of the rest of the network. Furthermore, costs are incurred due to switching equipment at each end. The problem of determining the cheapest network design is, in fact, one of enormous proportions. This difficulty is compounded by the need for any solution process to be able to interact with the requirements of the real world. It turns out to be impossible to achieve an exact solution to a real problem in anything approaching a reasonable amount of time.

A wide variety of different approaches have been suggested. Some make simplifications to the real case in order to allow established optimisation techniques to be used (such as assuming constant link costs and using the EMST). Others artificially restrict the number of possible solutions so as to decrease search time. The most effective techniques make no such assumptions, but instead apply a heuristic procedure which approximates the optimum solution to the real problem in an acceptable amount of time. Such methods, however, are not in abundance and those that do exist are generally in operation on machines with extremely large processing capabilities.

The purpose of this thesis, subsequent to discussing and attempting to extend the existing methods, is to propose a number of new, but still realistic, techniques for telecommunication network design. It is one of the major objectives of these methods that, in addition to providing acceptable levels of accuracy, the software used to implement them should be capable of running within the confined storage and processing environment of a small machine - a personal desktop computer for example.

1.1. Layout of the Thesis

The search for algorithms which solve various components of the network optimisation problem (NOP) requires a working knowledge of three distinct areas: the theory of algorithms, the principles of telecommunication networks and the underlying concepts behind networks - graph theory.

Chapter 2 discusses the important elements of graph theory, including the necessary definitions and some basic graph types. The more relevant graphs are enumerated and certain graphical optimisation problems are investigated.

Problem solving algorithms may differ considerably in detail. The important aspects which decide whether a particular algorithm is acceptable are discussed in chapter 3. This is concerned with the study of the classes P, NP and NP-complete.

Chapter 4 introduces the practical details of telecommunication networks and discusses some elementary concepts such as equipment costs, traffic principles, grades of service, etc.. The private automatic branch exchange (PABX) is then introduced. Its purpose is described in conjunction with the integrated services digital network (ISDN) and the chapter concludes with a discussion of PABX interconnection requirements.

The NOP itself is rigidly defined and discussed in chapter 5. Firstly the problem is specified precisely in terms of the required input and output to a solution algorithm. Its complexity is then analysed and the problems of using exhaustive search methods considered. The more significant existing methods of network optimisation are investigated and, wherever possible, these are extended and combined. Techniques are discussed for dealing with centralised networks and those with concentration facilities and the linear programming approach is considered. The chapter concludes by presenting a more practical method of solving the NOP based upon one of the simplified algorithms. The merits and drawbacks of this approach are noted.

A number of new techniques for solving parts of the NOP are presented in chapter 6. Firstly, a reduction process is introduced for replacing a large network problem with a smaller one which can be acted upon with less difficulty. The solution obtained in this way is translated into a solution for the original problem. A method of temporary tandem network topology restriction is then introduced in order to generate the first solution set of tandems. This solution can be acted upon by a perturbation algorithm in a search for improved solutions. The potentially massive problems of tandem network interconnection and traffic rerouteing are dealt with using a lookahead method of link removal once the tandems are fixed. A compound algorithm, comprising these and other techniques is then introduced and discussed.

Results of numerical experiments concerning real network problems are contained in chapter 7.

The techniques presented in chapter 6 are considered to be quite general. Chapter 8 investigates the extension of some of these techniques to other areas of problem solving. One such area is that of the Euclidean travelling salesman problem (ETSP) for which a new method of solution is suggested.

The concluding remarks on the NOP and the various methods of solution are to be found in chapter 9.

There are three appendices following the thesis. Appendix A contains the important algorithms and techniques not included in the main text, appendix B consists of traffic tables allowing for the calculation of link sizes and appendix C contains the user guide which accompanies the sofware package produced as a result of these investigations.

CHAPTER 2

GRAPH THEORY AND GRAPHICAL OPTIMISATION

A picture paints a thousand words

Anonymous

Graph Theory has its origins in such problems as that concerning the bridges of Königsberg (Fig 2.a). This asks whether it is possible to take a walk which includes the crossing of each bridge exactly once. An additional requirement may be that the walk start and finish from the same place. Examination shows this to be equivalent to asking whether the pattern in Fig 2.b can be drawn with a continuous movement of the pen (again with the possible requirement that the pen starts where it finishes).

This chapter is concerned with structures of the type shown in Fig 2.b, more commonly known as graphs. After the necessary definitions, the cardinality of the more important sets of graphs will be discussed (Graphical Enumeration) followed by certain problems in which an individual element of a set of graphs is requested (Graphical Optimisation).

The answer to both forms of the Königsberg bridge problem will be seen to be that no such walk exists (Biggs et al. [1976]). The general theory of graphs, however, extends far beyond this isolated example.

2.1. Definitions and Applications

This section prepares for a discussion of graphical enumeration and optimisation. Firstly, the essential components of a graph are described followed by the practical applications of such a model. The general theory of graphs is then extended as far as necessary before highlighting the concept of graphs in the 2-D Euclidean plane. These will prove to be the graphs of greatest importance.

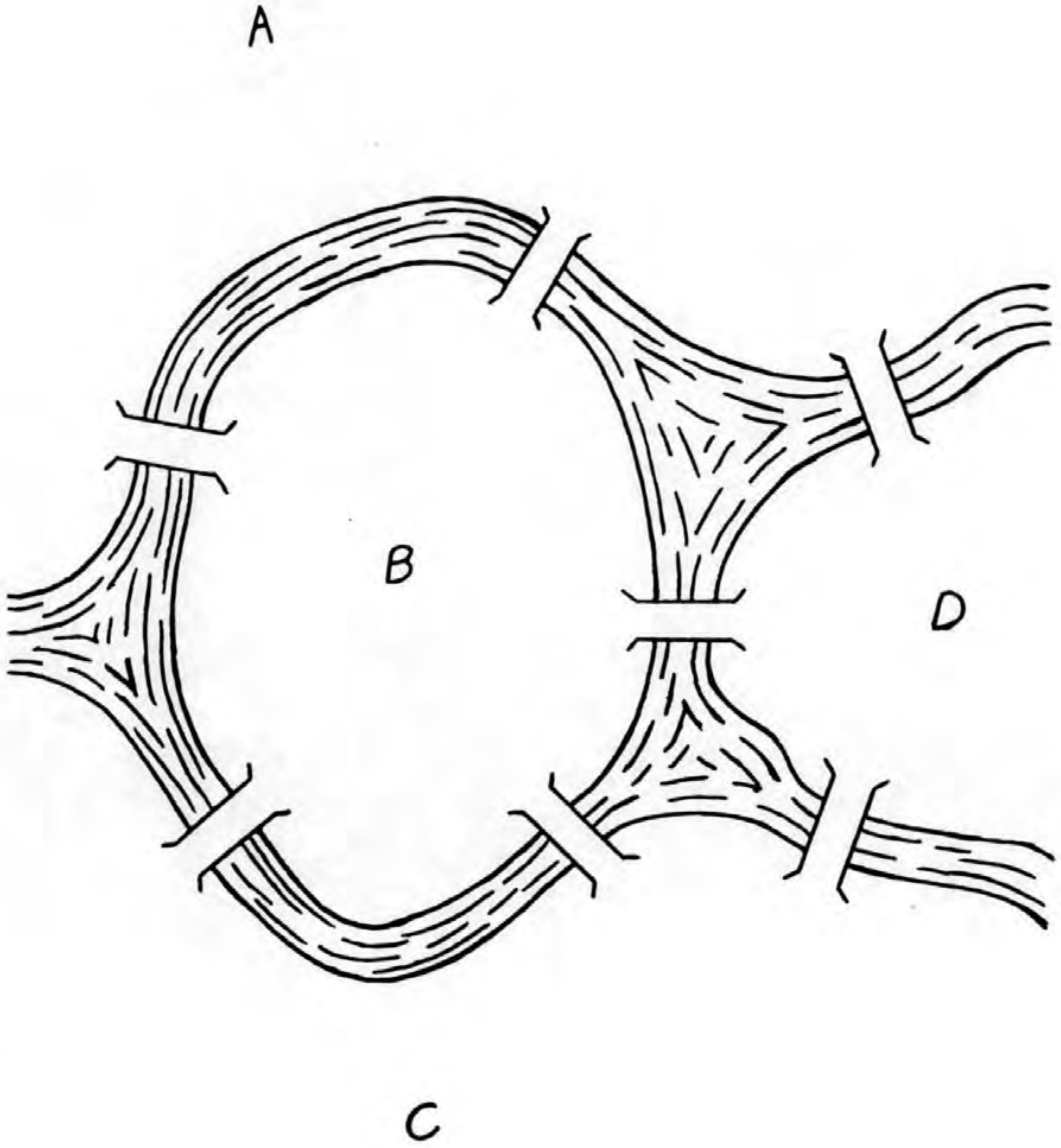


Figure 2.a. The Königsberg Bridge Problem

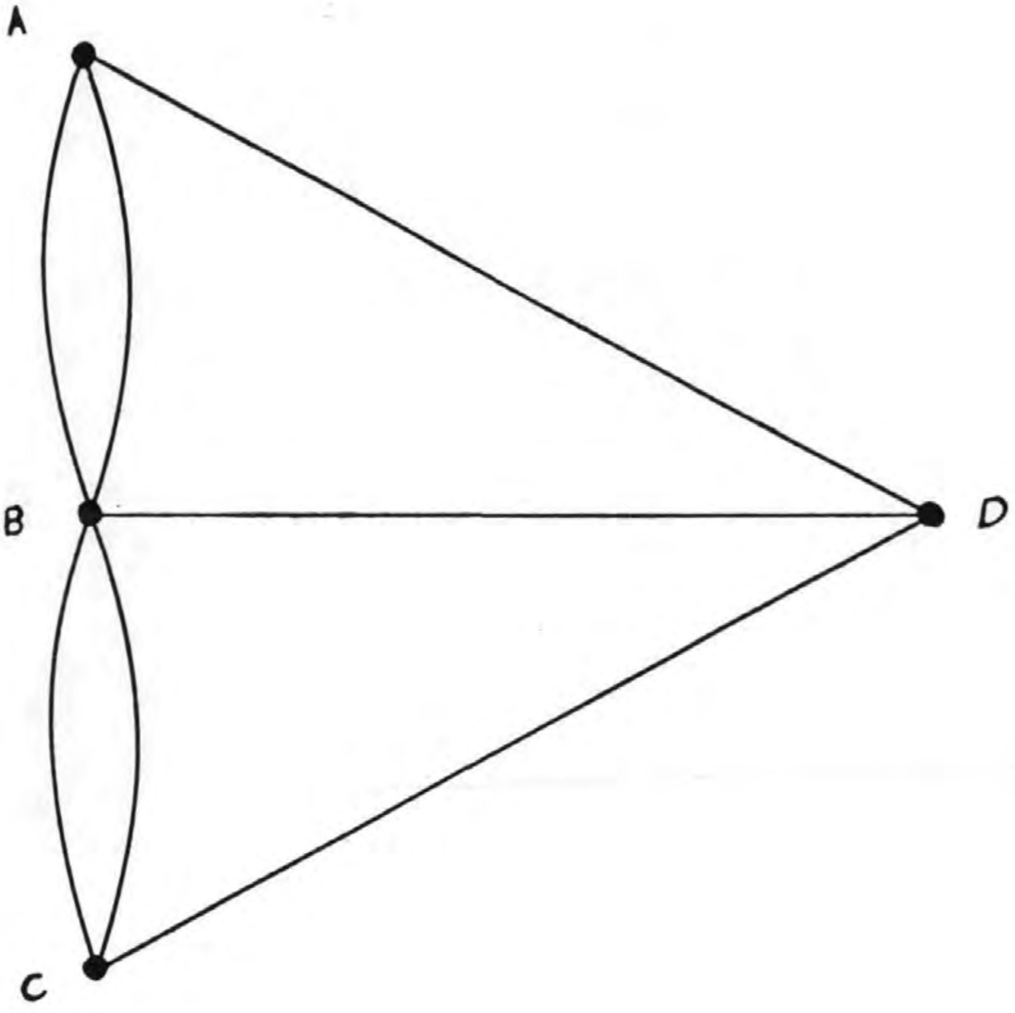


Figure 2.b. Graph of the Königsberg Bridge Problem

It should be noted that some considerable discord exists over graph notation and terminology. The following is adapted from the most common (Wilson [1972]) but it may not match that used by other authors on the subject.

2.1.1. Principles of Graphs

A graph G is formed by a pair (N,E). N is a non-empty and finite set of elements called nodes. E is a finite set of unordered pairs of elements of N called edges. Fig 2.c is a pictorial representation of a graph with N = {u,v,w,x} and E = {(v,w),(v,x),(w,x),(w,x),(x.x)}. In future, such a pictorial representation will not be distinguished from the graph itself.

If the (x,x) loop and the double edge (w,x) of Fig 2.c are not permitted, the result is a simple graph. In general, any reference to a graph in the text will imply a simple graph.

If E is made a set of ordered pairs of N then the graph becomes a directed graph or digraph and the elements of E are known as arcs. The set E is then usually renamed A.

The number of edges terminating at a node x (or adjacent to x) is the degree of x, denoted $\sigma(x)$.

Two graphs are isomorphic if there exists a bijection $f:N_1 \rightarrow N_2$ such that the edge (u,v) exists in E_1 iff (if and only if) the edge (f(u),f(v)) exists in E_2. A subgraph G_s of a graph G is a pair (N_s, E_s) such that N_s is contained in N and E_s is contained in E.

These are the simplest and most necessary definitions for dealing with graphs. Other important concepts will be introduced in the appropriate places.

2.1.2. Graphical Representation

The idea of a graph is an extremely powerful one. Many real situations can be modelled by such a structure and important deductions made on the basis of that which is already known about the underlying graph.

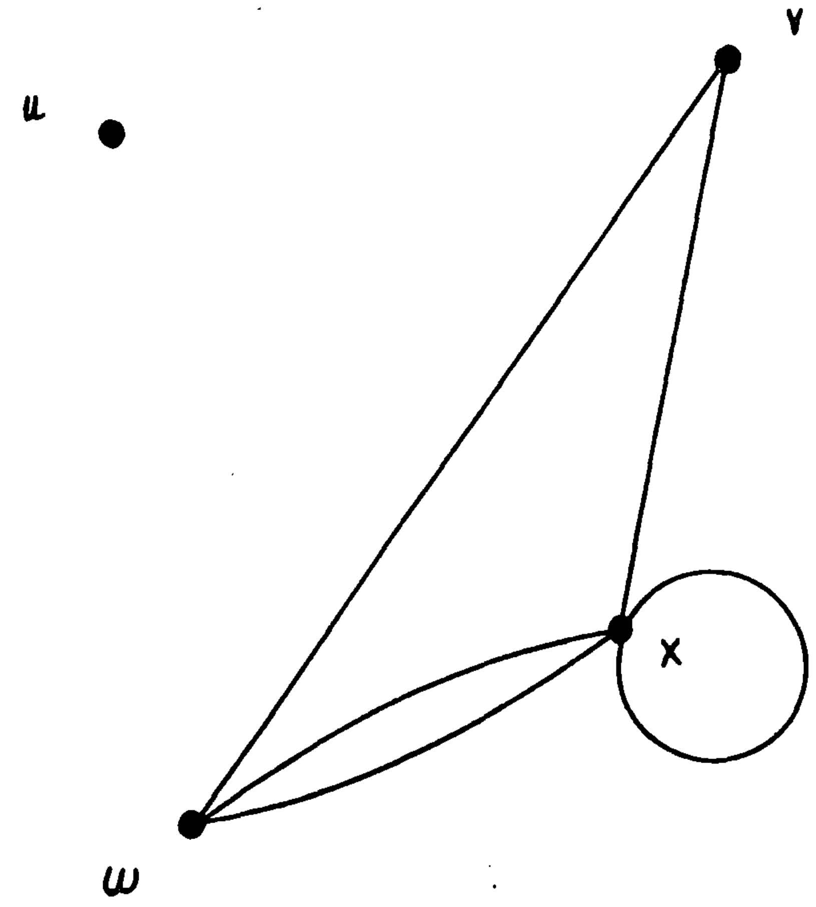


Figure 2.c. A Typical Graph

The most immediate examples are those things which, in some way, physically look like graphs. Towns and cities linked by rail and road networks, electrical circuits and components and water supply/removal facilities all have this obvious appearance.

However the power of graph theory can also applied to topics as diverse as sociology, geography, linguistics, operational research, numerical analysis, probability and many others (Bondy & Murty [1976], Deo [1974] and Wilson & Beineke [1979]). It is partly due to this wide field of relevance that so much has been achieved with the subject in a relatively short time.

The notion of a telecommunications network with senders and receivers of messages and transmission lines connecting them clearly belongs to the former category. This connection will be made more formal in Chapter 4.

2.1.3. Graph Theory

This section seeks to extend the basic ideas of section 2.2.1 in the directions relevant to the forthcoming chapters. For the purposes of this chapter, purely theoretical graphs are to be considered with little or no thought given to their application.

2.1.3.1. Further Definitions

A null graph or totally disconnected graph is one with an empty edge set E. The opposite is a complete graph or fully connected graph in which $(u,v) \in E$ for all $u,v \in N$ $(u <> v)$. The null and complete graphs on n nodes are denoted by N_n and K_n respectively.

A graph in which $\sigma(u)=\sigma(v)$ for all $u,v \in N$ is a regular graph of degree $\sigma(u)$. A complete graph is then clearly regular of degree n-1.

A bipartite graph is a graph in which the node set N can be split into two subsets N_1 and N_2 such that every edge of E joins a node in N_1 to a node in N_2. If every node in N_1 is connected to every node in N_2 then the graph is a complete bipartite graph, denoted by $K_{\alpha,\beta}$ where α and β are the cardinalities of N_1 and N_2.

A sequence of edges of E, $\{(u_1,u_2),(u_2,u_3), \ldots ,(u_{p-1},u_p)\}$ is called a path. If $u_1=u_p$ the path is closed and is known as a circuit. A path which includes every edge of E is called an Eulerian path and a path which visits every node in N is called a Hamiltonian path. Similarly, a circuit which includes every edge of E is an Eulerian circuit and one which visits every node in N is a Hamiltonian circuit or, more commonly, a tour.

If there exists a path from every node in a graph to every other node, the graph is said to be connected, otherwise it is disconnected. A comparable definition exists for digraphs which are said to be weakly connected if the underlying graph is connected and strongly connected if there exists a path between every pair of nodes which obeys the ordering of the edges in E. A subgraph G_s of G which is connected is a connected component of G.

A set of edges which, if removed, cause the graph to become disconnected is called a cutset. If this set consists of only one element, this edge is known as an isthmus.
A connected graph with $\sigma(u)=2$ for all $u \in N$ is a circuit graph and a connected graph with $\sigma(v)=n-1$ where $v \in N$ and $\sigma(u)=1$ for all $u \in N$ $(u<>v)$ is called a star.

A connected graph with n-1 edges is known as a tree. A tree connects a number of nodes with the fewest possible edges. A star is an example of a tree. If the edges of a graph are assigned numerical weights then a set of edges which connect the nodes with minimum weight sum is called a minimum spanning tree. The addition of one extra edge to any tree will create a graph with exactly one circuit.

2.1.3.2. Graph Networks

A digraph $D=(N,A)$ with weights Θ assigned to each arc $a \in A$ is called a network. $\Theta(a)$ is called the capacity of a. A network is then the pair $N=(D,\Theta)$ where D is the underlying digraph and $\Theta:A \rightarrow \mathbb{R}$ is the capacity function.

A flow in the network N is a function $\delta:A \rightarrow \mathbb{R}$ satisfying

(a). $\delta(a) \leq \Theta(a)$ for all $a \in A$

and (b). the total flow out of any node is equal to the total flow into the node.

If $\delta(a)=\theta(a)$ for any $a \in A$, a is said to be saturated. Many practical problems are concerned with maximising flows in networks.

In pure graph theory the definition of a network is extremely precise. It should be noted, however, that in the real world the term is often applied to quite different situations and often is meant simply to describe any interconnection structure, independent of flow.

2.1.3.3. Eulerian and Hamiltonian Graphs

Recall the Königsberg bridge problem which introduced the chapter. It can now be seen that the questions can be rephrased as asking whether the graph of Fig 2.b contains an Eulerian path or circuit. If so the graph would be said to be semi-Eulerian or Eulerian respectively.

In general a graph is Eulerian if $\sigma(u)$ is even for all $u \in N$ and semi-Eulerian if there are exactly two nodes of odd degree. This can be shown by induction on the number of edges (Harary [1971] and Wilson [1972]). Clearly the graph of the Königsberg bridge problem satisfies neither of these conditions.

A similar definition exists for those graphs containing Hamiltonian paths or circuits. These are called semi-Hamiltonian and Hamiltonian graphs respectively. Less is known about Hamiltonian graphs than Euclidean graphs. Most useful theorems state sufficient conditions for a graph to be Hamiltonian which are clearly not necessary (Dirac [1952], Ore [1960] and Posa[1962]).

The underlying problems behind detecting Hamiltonian circuits (tours) will be dealt with in Chapter 3 and techniques for finding the shortest one discussed in section 2.3.2 and Chapter 8.

2.1.4. Euclidean Graphs

Section 2.1.3 discussed assigning weights to edges of a graph with respect to finding the minimal spanning tree. In the general case these weights can be quite arbitrary. The weights may be thought of as given by the matrix $W=(w_{ij})$ where $w_{ij}=\infty$ implies that the link between i and j is undefined (not present in the graph).

Suppose, however, that the nodes of the graph are defined as distinct coordinate pairs $i=(x_i,y_i)$ $i=1,2, \ldots ,n$ and that the weights of each edge (i, j) are calculated as $w_{ij} = \sqrt{[(x_i-x_j)^2+(y_i,y_j)^2]}$. Then the graph is said to be <u>Euclidean</u>. There are some special properties of Euclidean graphs.

E1. $w_{ij}>=0$ for all i, j with equality iff $i=j$

E2. $w_{ij}=w_{ji}$ for all i, j

E3. $w_{ik}\leq w_{ij}+w_{jk}$ for all i, j, k

Euclidean graphs are those which most closely represent the majority of real situations. Distances are measured in exactly this way and the weight or <u>cost</u> of joining i to j or moving from i to j etc. is directly dependent on the distance between them. Euclidean graphs and their extensions will be of particular interest in the future although many of the arguments to be developed will apply also in the more general asymmetric, non-Euclidean case.

2.2. Counting Graphs

A number examples of different types of graphs have been encountered. The purpose of this section is to determine (precisely wherever possible) how many graphs of some certain forms there are. Such objectives require the techniques of Combinatorial Mathematics. An excellent introduction to this subject is to be found in Anderson [1974] while Scott [1971], Reingold et al. [1977] and Tucker [1980] contain more advanced information. Only those graphs of relevance to the work in hand are enumerated here. Details of many other applications can be found in Harary & Palmer [1973] and Robinson [1968] [1970]

It is also necessary to make a fundamental distinction between two types of graph or, more accurately, between two ways in which graphs can be considered. This is the difference between labelled and unlabelled graphs.

Consider the graphs in Fig 2.d. They are obviously isomorphic ($f(a)=d$, $f(b)=b$, $f(c)=c$ and $f(d)=a$) so in one sense can be considered as the same graph. However they are not really identical graphs since node a is connected to node b in the first but not in the second. Clearly, the way in which graphs are counted (type enumeration) will depend on whether the individual nodes are distinguishable from each other.

In a labelled graph each node has a unique identifier (label) $i=1,2,\ldots,n$ (n the number of nodes). Two graphs G_1 and G_2 are considered to be the same if and only if $N_1=N_2$ and $(u,v)\in E_1$ if and only if $(u,v)\in E_2$.

In an unlabelled graph no distinction is made between nodes. Two graphs are considered the same if and only if they are isomorphic.

Clearly there can never be fewer labelled graphs of a certain type than corresponding unlabelled graphs. In that which follows, all graphs will be assumed to be labelled unless otherwise stated. It will become clear that such graphs are more important in this context.

The simplest problem is that of determining how many graphs (of any type) it is possible to construct on n labelled points, denoted by G_n. There are $n(n-1)/2$ possible edges in a graph with each edge either present or absent in any particular graph. Therefore $G_n = 2^{n(n-1)/2}$. This will form the basis for much of what is to come.

2.2.1. Connected Graphs

It should be clear, even at this early stage, that connected graphs will be of great significance in the practical issues that are to follow. It is more than a matter of interest then to determine an expression for C_n, the number of connected labelled graphs on n nodes. Unfortunately no explicit formula exists. The following derivation of a recursive formula is adapted from Harary & Palmer [1973].

Define a rooted labelled graph as a labelled graph in which one node has been marked as special (a root). Denote the number of rooted labelled graphs on n nodes as R_n and the number of rooted labelled

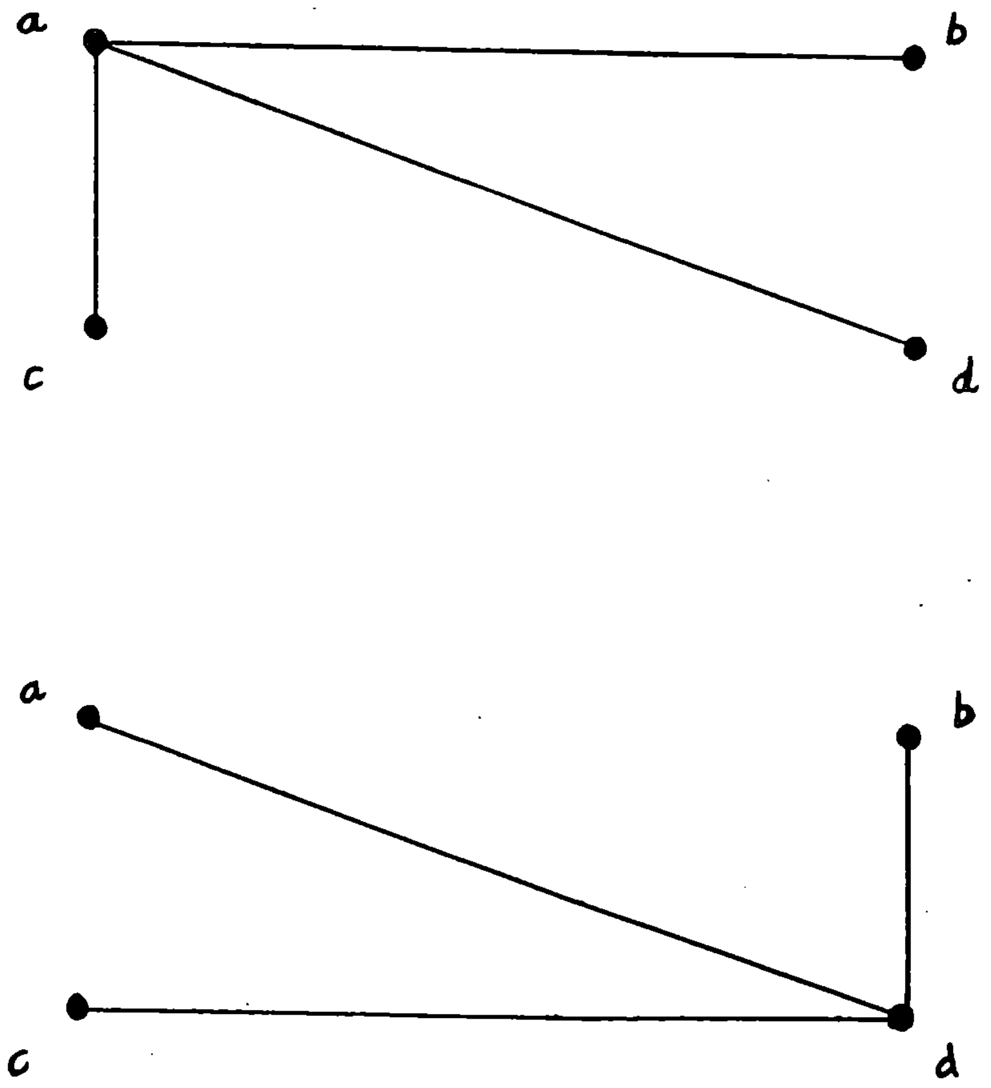


Figure 2.d. Isomorphic Graphs

graphs on n nodes whose root lies in a connected component of size i as R_{ni}.

A different rooted labelled graph is produced when a labelled graph (of any kind) is rooted at each of its nodes. Consequently

$$R_n = nG_n \quad \text{and} \quad C_n = R_{nn}/n = [R_n - \sum_{i=1}^{n-1} R_{ni}]/n$$

but consideration of the choice of subgraphs and roots shows that

$$R_{ni} = i\binom{n}{i}C_i G_{n-i}$$

so that $$C_n = G_n - \sum_{i=1}^{n-1} i\binom{n}{i}C_i G_{n-i}/n$$

$$C_n = 2^{n(n-1)/2} - \sum_{i=1}^{n-1} i\binom{n}{i}2^{(n-i)(n-i-1)/2}C_i/n$$

The first few values of C_n can be tabulated (Sloane [1973]) as

n	1	2	3	4	5	6	7	8	9
C_n	1	1	4	38	728	26,704	1,866,256	251,548,592	66,296,291,072

To demonstrate the point made in section 2.2, the corresponding values for connected unlabelled graphs are

n	1	2	3	4	5	6	6	8	9
C_n	1	1	2	6	21	112	853	11,117	261,080

A simple and convenient set of bounds for the number of labelled connected graphs is that C_n always lies between $2^{n(n-1)/2}$ and $2^{(n(n-1)/2)-1}$ ($C_n > G_n/2$ for $n \geq 4$).

2.2.2. Trees, Stars and Hierarchical Graphs

There are $T_n = n^{n-2}$ labelled trees which can be positioned on n nodes. This was originally shown by Cayley [1897] but the original proof has been adapted almost beyond recognition over the years. An outline proceeds as follows.

The assertion is proved by establishing a bijection between all labelled trees on n nodes and the set of all ordered strings $a_1a_2a_3$ .. a_{n-2}, $1 =< a_i =< n$ for all i. Clearly there are n^{n-2} such strings.

Consider any labelled tree and determine the lowest labelled vertex of degree 1, α_1 say. Let a_1 be the label of the only vertex adjacent to α_1. Now remove α_1 and its connecting isthmus. Repeat this until the graph consists of only two nodes and the required string $a_1a_2a_3$ .. a_{n-2} is determined.

The reverse process is equally simple. Take α_1 as the smallest label which does not appear in the string $a_1a_2a_3$ .. a_{n-2} and put in the edge between a_1 and α_1. Delete a_1 from the string and set $\alpha_1 = \infty$. If this process is repeated the required tree is produced.

Calculating the first few values of T_n gives the following table.

n	1	2	3	4	5	6	7	8	9
T_n	1	1	3	16	125	1,296	16,807	262,144	4,782,969

The corresponding table for the number of unlabelled trees is as follows

n	1	2	3	4	5	6	7	8	9
T_n	1	1	1	2	3	6	11	23	47

The situation with labelled star graphs is much simpler. Each node may be the centre of the star so the number of labelled star graphs on n nodes is $S_n = n$. By comparison, there is only one unlabelled star graph for each value of n.

A hierarchical star has the form shown in Fig 2.e. Again, a single node is chosen as centre with n such choices available. From the remaining n-1 nodes, a further m secondary centres are selected.

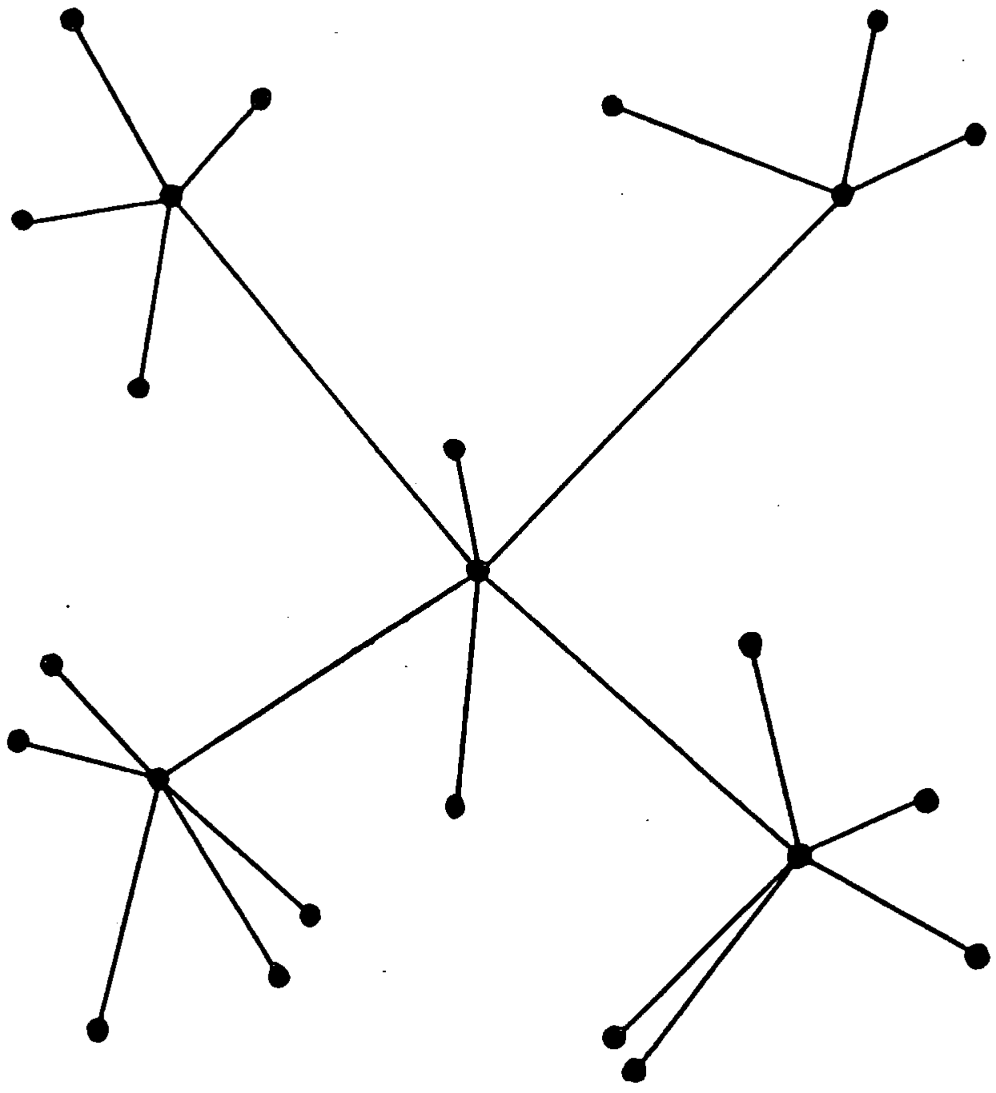


Figure 2.e. A Hierarchical Star

There are ${}^{n-1}C_m$ ways of doing this. Finally each of the n-m-1 nodes that are not centres or secondary centres can be connected to one of the m secondary centres or the centre. The number of <u>hierarchical m stars</u> which can be constructed on n nodes is therefore

$$HS_{n,m} = n\binom{n-1}{m}(m+1)^{(n-m-1)}$$

The first few approximate values of $HS_{n,m}$ are presented in the following table.

n	m 1	2	3	4	5	6	7	8
2	2							
3	12	3						
4	48	36	4					
5	160	270	80	5				
6	480	1620	960	150	6			
7	1340	8500	8960	2630	252	7		
8	3580	40800	71700	35000	6050	392	8	
9	9220	184000	516000	394000	109000	12300	576	9

If Euclidean graphs only are being considered then these numbers can be made smaller by insisting that each node be connected to the centre which is the shortest Euclidean distance away. If this is the case then $HS_{n,m} = n({}^{n-1}C_m)$ and the revised table is as follows.

n	m 1	2	3	4	5	6	7	8
2	2							
3	6	3						
4	12	12	4					
5	20	30	20	5				
6	30	60	60	30	6			
7	42	105	140	105	42	7		
8	56	168	280	280	168	56	8	
9	72	252	504	630	504	252	72	9

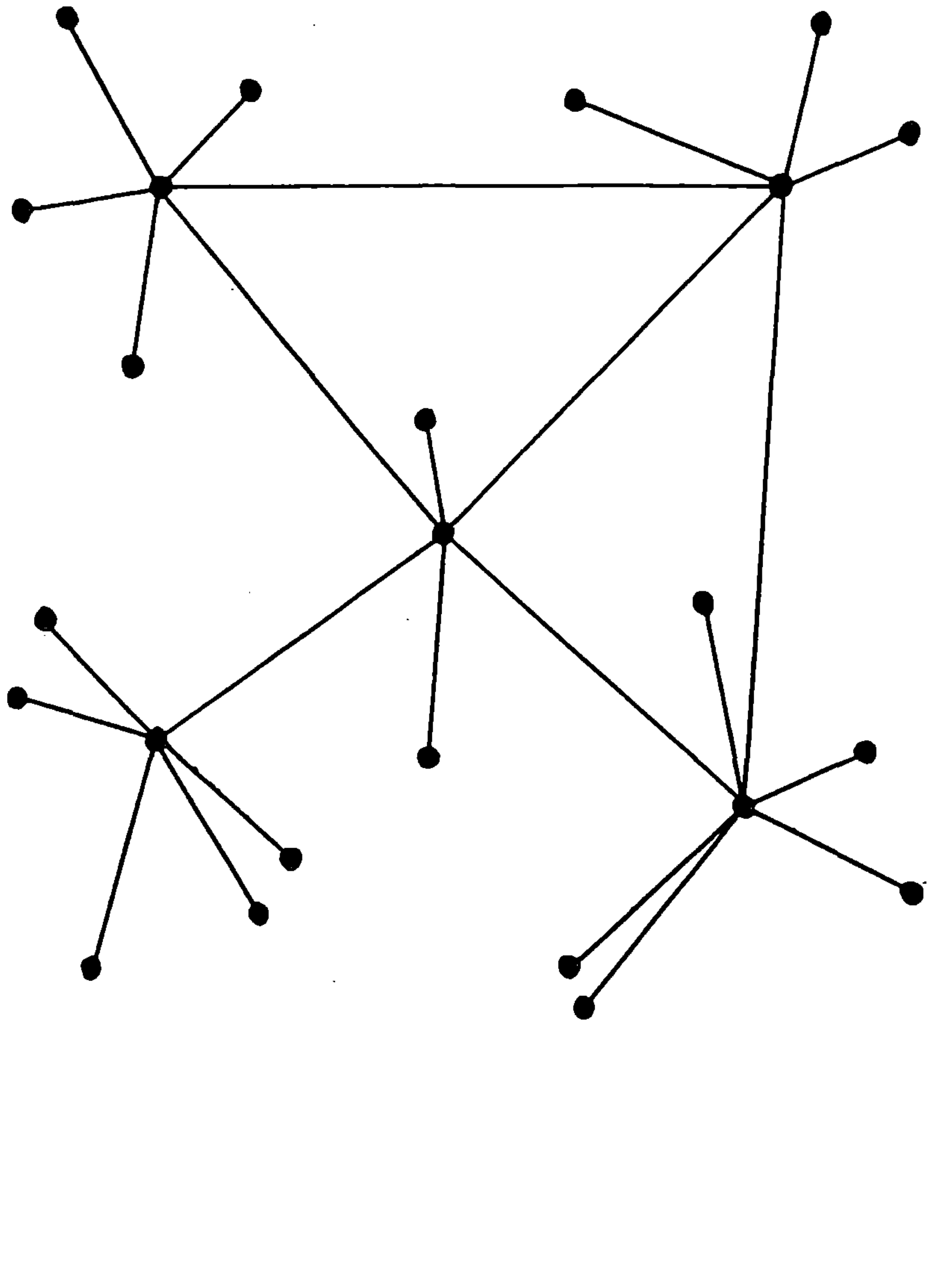


Figure 2.f. A Hierarchical Net

As with the case of the simple star, there is only one unlabelled hierarchical star for each pair of values (n,m).

A <u>hierarchical net</u> has the form shown in Fig 2.f. It is similar to a hierarchical star in that there are two levels to the graph. In this case, however, the m secondary centres form any connected graph instead of being directly connected to a centre. The number of <u>hierarchical m nets</u> which can be constructed on n nodes is given by

$$HN_{n,m} \;=\; \binom{n}{m} m^{n-m} C_m$$

The first few approximate values of $HN_{n,m}$ are presented in the following table.

n	m 1	2	3	4	5	6	7	8
1	1							
2	2	1						
3	3	6	4					
4	4	24	48	38				
5	5	80	360	760	728			
6	6	240	2160	9120	21800	26700		
7	7	672	11300	85100	382000	1120000	1870000	
8	8	1790	54400	681000	5100000	26900000	105000000	252000000

As before, these numbers can be made smaller by insisting that each node be connected to the centre which is the shortest Euclidean distance away. If this is done then $HN_{n,m} = (^nC_m)C_m$ and the revised table is as follows.

n	m 1	2	3	4	5	6	7	8
1	1							
2	2	1						
3	3	3	4					
4	4	6	16	38				
5	5	10	40	190	728			
6	6	15	80	570	4370	26700		

7	7	21	140	1330	15300	187000	1870000	
8	8	28	224	2660	40800	748000	14900000	252000000

The number of unlabelled hierarchical nets for each pair (n,m) is simply the number of unlabelled connected graphs which can be constructed on the m centres since the node to centre edges are indistinguishable.

2.2.3. Tours and Subtours

Section 2.1.3.3 introduced the concept of a Hamiltonian graph in which a tour was possible visiting every node of the graph exactly once. Determining whether a given graph is Hamiltonian is a difficult problem in the general case (Miller & Thatcher [1972] and Karp [1975][1976]). Determining the actual number of Hamiltonian circuits in a Hamiltonian graph is an extension which causes similar problems.

Consider, however, a special case. The complete graph on n nodes, K_n is clearly Hamiltonian. This can be seen by labelling each node and noting that the edges (1,2), (2,3), (3,4), .. (n-1,n) and (n,1) form a Hamilton circuit. Alternatively, the degree of each vertex (n-1) satisfies the conditions stated in Dirac[1952], Ore [1960] or Posa [1962].

For any indistinguishable set of n nodes there is only one tour. However the number of labelled tours on n nodes is equal to the number of cyclic permutations of the integers 1,2,3, .., n, which in turn is equal to (n-1)!. The importance of this figure will be considered in later chapters.

A natural relaxation is to remove the requirement that the permutation of the integers 1,2,3, .., n be cyclic. If this is the case the result is a set of <u>subtours</u> with each node included in exactly one subtour. The number of such sets of subtours is equal to the number of perturbations of the integers 1,2,3, .., n, i.e. n!.

The connection between tours and subtours will be exploited in Chapter 8.

2.3. Minimisation and Optimisation Problems

Section 2.2 was concerned with counting the number of graphs possessing a certain property. In all cases concerning labelled graphs and most concerning unlabelled graphs each set contained more than one member. In most cases a large number of graphs were included and this number tended to increase sharply with the number of nodes.

This section is concerned with selecting the individual graph, path, circuit, etc. which possesses some unique quality among all members of its set. Some of the sets considered will relate to those enumerated above while others will be new to the text.

These problems and results are a part of the much larger field of Combinatorial/Graphical Optimisation, good introductions to which can be found in Christofides et al. [1979] and Johnsonbaugh [1984] while more specialist areas are dealt with in Swamy & Thulasiraman [1981] and Lawler et al. [1985]. Some aspects are expanded in Chapter 8.

2.3.1. Shortest Connection Graphs

A tree connects n nodes together using n-1 edges. This is the minimum number of edges necessary to achieve this and a tree is the only type of graph with this property. An arbitrary tree can be constructed on a labelled set of n nodes by selecting any sequence of n-1 edges $e_1, e_2, \ldots, e_{n-1}$, subject to the restriction that each new edge e_i forms no circuit with any of the edges $e_1, e_2, \ldots, e_{i-1}$, already chosen.

If each edge e_i has a weight w_i associated with it then for any tree T a cost C(T) can be calculated as the sum of the weights of the edges in the tree. Of all the n^{n-2} trees which can be constructed on n nodes, at least one, T^* will have the property that $C(T^*) \leq C(T)$ for all trees T. A number of practical applications, particularly in the field of operations research, require that this T^* be found.

The minimum spanning tree problem is well solved. The following simple algorithm is known as Kruskal's algorithm (Kruskal [1956]). It is not alone in finding T^*. An algorithm of Prim [1957] and Dijkstra

[1959] is almost as straightfoward and, in fact, gives improved computational performance.

K1. Construct a sequence of n-1 unique edges $e_1, e_2, \ldots, e_{n-1}$ by choosing each new edge e_i as described in K2.

K2. e_i is the unchosen edge of least weight w_i which forms no circuit with edges $e_1, e_2, \ldots, e_{i-1}$ already chosen.

It is clear that such a method will produce a tree and that the constraints are necessary to produce the minimum. It is not as clear that they are also sufficient.

Suppose, to the contrary, that there exists a tree T' of smaller cost than T^*, the tree produced by Kruskal's algorithm. Clearly at least one edge in the sequence $e_1, e_2, \ldots, e_{n-1}$ is not present in T'. Let the <u>first</u> such edge be e^*. Add e^* to T' to produce a new graph S. This graph contains a unique circuit (see section 2.1.3.1). Within this circuit must lie an edge e' which is contained in T' but not in T^*. Replace e' by e^* in T' to form a new tree T". Then $C(T'') \le C(T')$ since $w^* \le w'$ by the choice of e^* and T" is one edge closer to T^* than T'. Repeating this action produces a sequence of trees T', T", .., T^* with the property $C(T') \ge C(T'') \ge \ldots \ge C(T^*)$, contradicting the original supposition. Hence T^*, as produced by Kruskal's algorithm, is a minimum spanning tree.

2.3.2. Paths, Circuits and Tours

Three classical problems are discussed briefly in this section: finding the shortest path between two nodes in a graph, determining an Eulerian circuit and obtaining the shortest (optimum) Hamiltonian tour.

2.3.2.1. The Shortest Path Problem

Take a connected graph G and assign weights w_{ij} to each edge $(i,j) \in E$. Assign $w_{ij} = \infty$ to those edges not in E. For any two non-adjacent nodes u and v, it may be required to know the path $(u,u_1),(u_1,u_2),\ldots,(u_j,v)$ from u to v such that the sum of the weights of the edges used is a minimum.

Considerable work has been done on the shortest path problem, particularly with applications in operations research (Hu [1968] and Johnson [1977]). The following is a brief description of an algorithm due to Dijkstra [1959]. It involves working across the graph, assigning a value $\mu(z)$ to each node z, until node v is reached. $\mu(z)$ represents the shortest path to the node z, calculated on the basis of the known shortest paths.

D1. Start at the node u. Include u in a set Γ. $\mu(u)=0$

D2. Find the node z, not in Γ, such that w_{xz} is a minimum for some $x\in\Gamma$.

D3. Calculate $\mu(z)$ as the minimum of $\mu(y)+w_{yz}$ for all $y\in\Gamma$.

D4. Include z in Γ. If z=v then stop else goto D2.

2.3.2.2. Finding an Eulerian Circuit

A simple test can be used to determine whether a given graph is Eulerian or Semi-Eulerian (section 2.1.3.3). Given such a graph, an equally straightforward algorithm can be used to generate a suitable Eulerian circuit or path. This is as follows.

E1. If the graph is semi-Eulerian then start at one of the nodes of odd degree. Otherwise start at any node.

E2. Construct any path or circuit subject to the condition in E3. As each edge is used, delete it along with any nodes isolated by this action.

E3. At each stage, choose the new edge to be an isthmus only if there is no other choice.

Similar ideas are encountered in the study of the Rural Postman Problem (Orloff [1974]) and Chinese Postmen Problem (Edmonds [1965]). The Rural Postman Problem requires the shortest walk to be found which includes every edge of the graph. Clearly if the graph is Eulerian then an Eulerian circuit, as generated above, will provide a solution. If the graph is non-Eulerian, some edges will need to be used twice and the problem becomes more involved. The Chinese Postman Problem is a generalisation of this in which digraphs or mixed graphs with regular and directed edges are used.

Many variations of these postal problems are well solved (Edmonds & Johnson [1973]) but the most general case is known to be hard.

2.3.2.3. The Travelling Salesman Problem

For a given Hamiltonian Graph there may be more than one acceptable tour. Indeed for K_n there are $(n-1)!$. The Travelling Salesman Problem (TSP) requires that the tour of minimum cost be found.

Suppose the cost of moving from node i to node j in K_n is given by c_{ij}. The TSP then requires that a cyclic permutation π of the integers 1, 2, .., n be found which minimises

$$z = \sum_{i=1}^{n} c_{i\pi(i)}$$

Probably no other problem in optimisation theory has attracted the quantity of work which the TSP has. Many algorithms exist for its solution. Some are exact (Held & Karp [1962], Bellman [1962] and Volgenant & Jonker [1982]) while others are approximate (Karg & Thompson [1964], Shapiro [1966], Lin & Kernighan [1973], Christofides [1976] and Grout, Sanders & Stockel [1987b]). Many use established mathematical techniques (Dantzig, Fulkerson & Johnson [1959], Golden [1977] and Balas & Christofides [1981]) while some methods appear to have been developed solely for that purpose (Van Der Cruyssen & Rijckaert [1978] and Parker & Rardin [1984]). Probably its interest lies, not in any enormous applicability to everyday problems, but in its almost perfect representation of all the problems of combinatorial optimisation. Its solution, for other than trivial or special cases, is known to belong to a class of extremely difficult problems. These problems will be encountered in the next chapter while the TSP will be discussed in greater detail in chapter 8.

2.3.3. Maximum Flow in a Network.

Section 2.1.3.2 described a network and the concept of a flow within a network. Given a network $N=(D,\Theta)$ and two distinct nodes u and v, it is often required to find the maximum flow possible from u (known as a <u>source</u>) to v (<u>sink</u>). This problem has important applications in traffic flow calculations (Flood [1975]) and product distribution.

A Theorem of considerable importance in this context is known as the <u>Maximum Flow Minimum Cut Theorem</u> (Ford & Fulkerson [1956] and Elias et al. [1956]). This states that a maximal flow can always be found equal to the cardinality of the smallest cutset disconnecting u and v. A flow which satisfies this principle can be found via the following algorithm.

F1. Find, by any method, a valid non-zero flow δ.

F2. If δ satisfies the conditions of the maximal flow minimal cut theorem then stop.

F3. Reverse the direction of the flow δ in N to produce a new network N' (i.e. an unused arc in N is unchanged in N' whereas any used arc a is replaced by an arc with capacity $\Theta(a)-\delta(a)$ and an arc in the opposite direction with capacity $\delta(a)$).

F4. Find, by any method, a valid non-zero flow Γ in the network N'. Set $\delta(a)=\delta(a)+\Gamma(a)$ for all arcs.

F5. Goto F2.

CHAPTER 3

COMPUTATIONAL COMPLEXITY OF PROBLEM SOLVING ALGORITHMS

L'embarras des richesses
(The more alternatives, the more difficult the choice)
Abbe D'Allainval, 1726

Any machine which is capable of providing computation does so by being able to perform a finite number of operations. If necessary, any discussion of algorithms could be made relative to the simplest concepts of Turing Machines (Brady [1977]) or Unlimited Register Machines (Cutland [1980]) in which the set of such operations is both small and simple. In practice, however, it is more convenient to deal with machines of greater power. This improved approach can be shown to be equivalent to the original one (Aho et al. [1974]) in the sense that the sets of problems capable of being solved in each case are identical.

In addition it is useful to introduce the concept of a language. A language is, in this context, a means of communication between the user and the machine consisting of a set of steps in which a single step comprises a number of basic operations (+, -, /, etc.) together with a control structure to allow the steps to be carried out in the correct order. The language used here will be loosely based on Pascal (Hoare & Wirth [1972] and Findlay & Watt [1978]) but similar arguments could be presented in most high level languages (Balfour & Marwick [1979] and Kelly & Pohl [1984]).

An algorithm is an ordered series of steps which can be expressed in any particular language. Included in the algorithm may be control statements (for .., while .., etc.). This determines the order in which the steps of the algorithm are carried out. The normal flow is sequential unless directed otherwise by a control statement.

To operate, an algorithm requires a set of input parameters $I:=\{i_1,i_2,..,i_\alpha\}$. It will generate a set of output parameters

$O=\{o_1,o_2,..,o_\beta\}$. The set O will be dependent upon the set I, as, in some cases, will be the actual operation of the algorithm.

3.1. Counting Steps in Algorithms

Languages differ in their definition of a step. Some may allow exponentation, for example, whereas others will require that such an operation is carried out as a sequence of multiplications. In general then, two algorithms to solve the same problem in two different languages will take an unequal number of steps. This apparently presents difficulties to the analysis of algorithms in that a problem A could be solved faster than a problem B in a language L_1 but slower in a language L_2. Closer inspection, however, shows that the implementation at machine level is generally the same.

Section 3.2 will discuss a fundamental distinction between two classes of problem. It will transpire that this distinction is so powerful that, for all practical purposes, the above dilemma matters little, even in pure algorithmic terms. The remainder of this section is concerned with the methods used to describe the running time of different algorithms in a uniform way.

3.1.1. Deterministic Algorithms

A <u>deterministic algorithm</u> is a sequence of steps as defined above consisting of <u>deterministic statements</u>. The outcome of such a statement may depend on a set of parameters $P=\{p_1,p_2,..,p_\pi\}$, drawn from the input parameter set I and the temporary algorithm parameter set $T=\{t_1,t_2,..,t_\tau\}$. Two identical parameter sets P_1 and P_2 always produce the same outcome from the same statement, with no variation or random decision feature. The algorithm is thus deterministic in all of its decision making processes.

All existing operational languages, whether high or low level, restrict the programmer to the use of deterministic algorithms. An {if $B(p_1,p_2,..,p_\pi)$ then} statement, where B is a Boolean function, will always give the same result if the values of $p_1,p_2,..,p_\pi$ remain unchanged since B will always take the same value of TRUE or FALSE.

3.1.2. Counting Steps

Counting the number of steps in a simple algorithm with no control statements is trivial. Consider, however, the following algorithm which accepts the coordinates of n cities as input and calculates the distance between each pair.

```
$I=\{x_1,y_1,x_2,y_2,\ldots,x_n,y_n\}$
for i := 1 to n do
    for j := 1 to n do
        if i=j then $d_{ij}$ := 0
                else $d_{ij}$ := $[(x_i-x_j)^2+(y_i-y_j)^2]^{1/2}$
$O=\{d_{11},d_{12},d_{13},\ldots,d_{ij},\ldots,d_{nn}\}$
```

Within the nested loop, exactly one of the steps is executed. The innermost part of the loop is encountered n x n times. The running time of this algorithm is therefore n^2 steps. This simple approach can be extended to more sophisticated algorithms without much difficulty.

One slight problem exists. Often the actual flow of control within an algorithm will depend on the parameter set P, which ultimately depends on the input parameter set I. It may be impossible to trace through all stages of the algorithm, predicting the action taken at each stage. Under these circumstances it is customary to use a technique known as worst case analysis in which all steps are counted, even where their eventual execution is in doubt. There are drawbacks to this technique in that the final figure obtained may far exceed the actual number of steps taken by the algorithm. At least, however, it does provide a valid upper bound.

3.1.3. The O(f(n)) Notation

The process of counting steps in algorithms can be made more efficient in terms of practical use. In general, the running time of an algorithm with n input parameters will be a function g(n). For example the following algorithm accepts the coordinates and county identifiers of n cities as input and determines, for each pair, the distance between them as well as whether they belong to the same county. The 'distance' from each city to itself is set to infinity. This algorithm has running time $g(n)=2n^2+n$.

```
I={$x_1,y_1,c_1,x_2,y_2,c_2,...,x_n,y_n,c_n$}
for i := 1 to n do
    for j := 1 to n do
        if $c_i=c_j$ then $S_{ij}$ := TRUE
                 else $S_{ij}$ := FALSE
for i := 1 to n do
    for j := 1 to n do $d_{ij}$ := $((x_i-x_j)^2+(y_i-y_j)^2)^{1/2}$
for i := 1 to n do $d_{ii}$ := $\infty$
O={$d_{11},d_{12},d_{13},...,d_{ij},...,d_{nn},S_{11},S_{12},S_{13},...,S_{ij},...,S_{nn}$}
```

Now define a function $f(n)$ to be the single dominant factor of $g(n)$ with any multiplicative factor removed. In the above example, therefore, $f(n)=n^2$. The running time of the algorithm is said to be <u>order of</u> n^2 steps, written $O(n^2)$. The algorithm of section 3.1.2 also has $O(n^2)$ running time.

The O notation provides a convenient method for comparing algorithms. If algorithm A runs in $O(f_A(n))$ steps and algorithm B in $O(f_B(n))$ steps and $f_A(n)<f_B(n)$ then there is some criterion for saying that algorithm A is better than algorithm B.

The absence of any lesser term or multiplicative factor in the O(..) representation means that the slight variation in facilities provided by different languages and the different speeds of various machines do not affect this measure of running time of an algorithm. Another way of describing the O notation is to say that an algorithm of $O(f(n))$ steps will run in no more than $Cf(n)$ seconds where C is a constant dependent on the language and machine being used.

3.1.4. Non-Deterministic Algorithms

Suppose that, in addition to the commands and statements available for deterministic algorithms, a number of <u>non-deterministic statements</u> exist. Such statements are of the form either {statement_1} or {statement_2}

and it is expected that each time such a statement is encountered, exactly one of statement_1 or statement_2 will be executed, the correct choice being made inspirationally on each occasion. This should not be confused with the standard IF ... THEN ... ELSE ... statement of a deterministic algorithm. For example, the following non-deterministic algorithm solves the knapsack problem (Beker & Piper [1982]) for a set of n integers. It has running time of O(n) steps.

```
I:={$a_1, a_2, \ldots, a_n, \beta$}
$\alpha$ := 0
for i := 1 to n do
    either $\alpha := \alpha + a_i$ or (no action)
if $\alpha = \beta$ then Exist := TRUE else Exist := FALSE
O:={Exist}
```

Clearly no known real language has this non-deterministic ability. While it might be possible to introduce a level of randomness into the operation, there is then no guarantee that the correct decision would be made. Similarly no machine exists which can search through potentially unlimited numbers of {either {statement_1} or {statement_2}} branches to find the correct choice in the same time as an ordinary machine could execute the single step (although the advent of parallel processing has made this fact somewhat less obvious (Sumner [1986])).

Non-deterministic principles are, nonetheless, of considerable use in the development of the theory of algorithms. As will be seen in section 3.2, the concepts of deterministic and non-deterministic algorithms provide the basis of a method for classifying problems as easy or hard.

3.2. Complexity Classes of Optimisation Problems

It is known that there are some problems, well defined and easily stated, for which no algorithm can be written to provide a solution

(Turing [1936]). Such matters are not of concern here. For the purposes of this work, a problem P, of size n, is solved by an algorithm A. A runs in $O(f_A(n))$ steps. It is often of extreme importance to choose A such that $f_A(n)$ is a minimum. If such an algorithm can be found then the value of $f_A(n)$ gives a measure of the difficulty in solving P.

Some problems are harder than others. A problem for which the best algorithm runs in $O(2^n)$ steps is more difficult to solve than an problem for which the best algorithm runs in $O(n^2)$ steps. A useful, but apparently arbitrary, distinction among problems is to separate those which can be solved by an algorithm running in a polynomial number of steps from those which apparently cannot. For these purposes, orders such as $O(n^x)$, $O(n^x \log(n))$, etc. are regarded as polynomial whereas $O(x^n)$, $O(n!)$, $O(n^{\log(n)})$, etc. are considered exponential.

3.2.1. The Class P

If a problem can be solved by a deterministic algorithm running in polynomial time ($O(n^3)$, $O(n^{10})$, etc.) it is said to belong to the class P. This class represents the easiest of all problems with respect to running time.

In principle, at least, any problem in the class P can be solved for a large number of input parameters (n). Difficulties can arise with running times such as $O(n^{60})$ but in practice, such situations rarely occur.

Problems known to be in the class P include that of the Minimum Spanning Tree (Kruskal [1956], Prim [1957] and Dijkstra [1959]) and Linear Programming (Dantzig [1963] and Karmarkar [1984]).

3.2.2. The Class NP

A problem which can be solved by a non-deterministic algorithm running in polynomial time is said to belong to the class NP. Clearly any problem in P is also in NP since a deterministic algorithm is a valid non-deterministic algorithm which does not use the non-deterministic statements.

Trivially then P is at least contained in NP. It not known whether P=NP or P<>NP. No problem in NP has been found to date for which it has been proved that no deterministic polynomial time algorithm exists. On the other hand, however, a number of problems are known for which no such algorithm has been found. It is widely believed that P<>NP and much work has been done on the basis of this assumption (Beker & Piper [1982]).

3.2.3. NP-Complete Problems

A large number of the problems in NP for which no deterministic polynomial time algorithm is known have been shown to be equivalent to one another (Karp [1972] and Garey & Johnson [1979]) in the sense that any polynomial algorithm developed for one of them could be modified to provide a solution for each of the others in turn. These problems are said to be NP-complete. They represent the hardest of all solvable problems (with the dubious exception of the D^P and D^P-complete problems discussed by Johnson & Papadimitriou [1985]).

The overall structure of the complexity class system is believed to have P and NP-complete as disjoint subclasses of NP. There may or may not be problems in NP-P-(NP-complete).

Any newly encountered problem found to be in NP might be found to be in P. Failing this, it might be proved to be NP-complete by demonstrating its equivalence to one of the problems which are known to be NP-complete. Otherwise it remains unplaced in NP until either its NP-completeness is proved or a deterministic polynomial time algorithm is found (if either is possible).

3.2.4. Algorithms for Solving Large NP-Complete Problems

Any problem with a practical application will need to be solved using a deterministic algorithm since these relate to the facilities provided by actual machines and languages.

The running time of algorithms requiring an exponential number of steps will increase far quicker than those requiring a polynomial number of steps. If the problems to be solved are known to be large then algorithms of the latter type would be desirable. Such an algorithm only exists, however, if the problem is known to be in the class P. Many practical problems are not known to be in this class. Some indeed are known to be NP-complete, implying, if general belief is correct, that no such algorithm can exist.

This causes severe difficulties in solving very large NP-complete problems. The only escape is to use a polynomial time algorithm which provides an approximation to the correct solution. The value of this approximation may be ensured by either a guarantee that the solution is exact at least a certain percentage of the time or a guarantee that the approximation is always within a certain percentage of the correct solution. Some situations may allow both.

Practical cases of this type are discussed in chapters 5 and 6.

CHAPTER 4

TELECOMMUNICATION NETWORKS

He seems so near and yet so far
Alfred, Lord Tennyson
In Memoriam, 1850

Telecommunication networks of varying degrees of sophistication have existed for over 100 years. In general, the huge nationwide structures in use today have evolved from others already in operation at various stages throughout history (Murray [1982], Garbutt [1985], Muir & Hart [1987] and Ozdamar [1987]). Often the choice of sites and communication routes has been dictated by the existing equipment already available. Such networks may tend toward structures which are far removed from the most efficient and optimisation can be of little use in these cases. A recent design study (BP [1984]) recommended eight switching centres (or zone centres) for the network under consideration whereas earlier investigations had indicated that just three would be some £300 000 less expensive! A clear understanding of the structure of such networks can be useful with regard to possible alterations to be made in the future but, in general, the existing topology will be of more importance. In addition, other constraints such as alternate routeing may be in evidence and this complicates the situation even further.

In certain cases, however, networks do not evolve in this way. The trend towards private communication networks (Lane [1984] and Boardman & Fickling [1987]) and the emerging digital networks with new facilities (Heap & Arthur [1985] and Brown & Lidbetter [1985]) is producing a number of entirely new structures. In other countries as well, new networks are appearing as opposed to growing out of old ones (Upton [1987]). It is in these situations that optimisation is of most importance. With networks costing millions of pounds, a saving of even a few percent can easily justify a considerable amount of optimisation, in addition to the usual design work, before construction begins.

This chapter outlines the principles behind telecommunication networks. It deals with the theoretical aspects, the individual components of such structures, the facilities provided by these components and the overall layout, both at the present time and as predicted into the future.

4.1. Basic Principles and Design Constraints

A network of size n consists of n nodes; $i=1,2,..,n$ each defined by its coordinates (x_i, y_i) in the Euclidean plane. The purpose of the network is to allow pairs of nodes to communicate information between them. This information is known as traffic and is usually measured in erlangs (named after the Danish mathematician, A.K. Erlang (Brockmeyer et al. [1948])). If the mean number of calls from node i to node j in time t is c and the average length of each call (the holding time) is l then the number of erlangs from node i to node j is given by

$$a_{ij} = \frac{cl}{t}$$

One erlang can be considered as equivalent to an average of one call continuously. An alternative measurement, used primarily in the USA, is that of hundred call seconds per hour (ccs h^{-1}) where clearly one erlang is equivalent to 36 ccs h^{-1}.

Traffic can take a number of forms (voice, data, text, etc). In a network with nodes generating T types (T>1) it may be necessary to distinguish between them for each node pair, for the purposes of call logging, charging, etc.. A natural extension to the above notation is to let $a_{\tau ij}$, $i,j=1,2,..,n$, $\tau=1,2,..,T$ be the amount of traffic of type τ generated at node i and destined for node j and redefine a_{ij} as

$$a_{ij} = \sum_{\tau=1}^{T} a_{\tau ij}$$

The value $a_{\tau ij}$ can be regarded as the element in the ith row and jth column of an $n \times n$ matrix A_τ. The coordinates (x_i, y_i), $i=1,2,..,n$ and the T traffic matrices $A_\tau = (a_{\tau ij})$, $i,j=1,2,..,n$, $\tau=1,2,..,T$ describe the physical setup of the network and will comprise most of the input data to any network optimisation problem (NOP).

4.1.1. Network Performance Constraints

In addition to the node locations and traffic flow, a number of other elements of information are required as input to an NOP. Some of these place restrictions on the type of solution which is acceptable and constraints on the quality of performance provided by the network. Others merely represent the cost of various pieces of equipment or the period of time for which the network is intended to operate. The more important of these are mentioned in sections 4.1.1.1-4.

4.1.1.1. Network Structure

If the nodes of a network N can be split into two groups N_1 and N_2 such that $a_{ij}=a_{ji}=0$ for all $i \in N_1$, $j \in N_2$ then N_1 and N_2 can be treated individually as separate networks. If this is not the case then the first requirement of any solution network is that the underlying graph be connected. As seen in chapter 2 there are between $2^{(n(n-1)/2)-1}$ and $2^{n(n-1)/2}$ connected graphs on n nodes.

It is usual to specify further the topology of the solution network. The particular form, however, will depend upon the application for which the network is required. Obvious examples include the star and hierarchical star and net introduced in chapter 2 or a full mesh arrangement in which each node is connected to every one of the others.

Another example, usually applied to data transmission, is that of centralised networks (Esau & Williams [1966], Kershenbaum & Chou [1974], Chandy & Russell [1972] and Schwartz [1977]). Such topologies occur when a number of nodes require communication only with a central point ($a_{ij}=0$ if $i>1$ and $j>1$). This situation produces networks of the form shown in Fig 4.a. A large amount of work has been done with

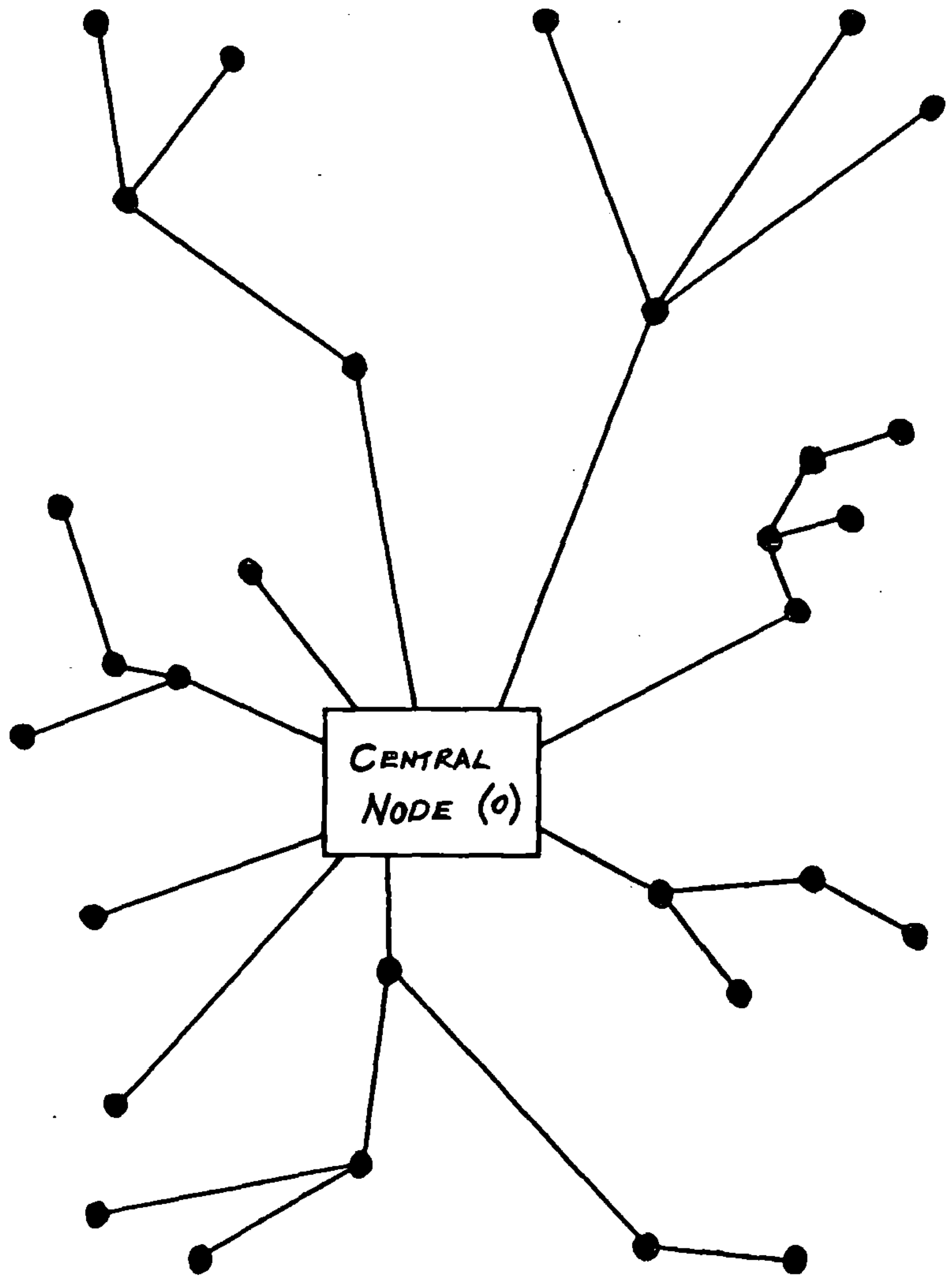


Figure 4.a. A Centralised Network

respect to these structures. In addition to the above references, methods have been suggested by Martin [1967], Sharma & El-Bardai [1970] and Elias & Ferguson [1974]. Such networks are covered in greater detail in chapter 5. They are less acceptable when communication is required between two non-central nodes as in the general case.

Practical network topologies for general telecommunication requirements will be discussed in section 4.3.

4.1.1.2. Grade of Service

If there are ∂ usable communication routes on a link between two nodes i and j then clearly the link can carry ∂ calls simultaneously. If a $\partial+1$th call is attempted before any of the other routes are released then the call will fail. The grade of service (GOS) of any link is the probability of this failure occurring.

A single direct communications route on a link between two nodes is called a circuit. If a link consists of c circuits and is required to carry A erlangs of traffic, the GOS can be approximated as follows.

If it is assumed that the number of possible call sources is large, that calls are attempted and terminated at random and that blocked calls are lost (as opposed to delayed) then it can be shown (Bear [1976]) that the probability of k circuits being busy at any time is given by

$$P(k) = \frac{A^k/k!}{\sum_{x=0}^{c} A^x/x!}$$

This is the Erlang B equation. Blocking occurs when k=c. The GOS for a link of c circuits carrying A erlangs of traffic is therefore

$$G(c,A) = \frac{A^c/c!}{\sum_{x=0}^{c} A^x/x!}$$

This is the <u>Erlang formula of the first kind</u>. Unfortunately, no explicit equation of the form $c = f(G,A)$ exists to determine the number of circuits required to carry A erlangs at GOS G. Substantial tabulation, however, does exist and some is presented in appendix B.

In general a required GOS will be part of the input to any NOP. There are, however, two types of GOS. The Erlang equation calculates the <u>link GOS</u>. An alternative is the <u>end to end GOS</u>. If the path between two nodes i and j consists of l links with link grades of service $G_1, G_2, \ldots, G_l$ then the end to end GOS between i and j is given by

$$E = 1 - \prod_{\mu=1}^{l} (1-G_\mu)$$

$$= 1 - (1-G_1)(1-G_2)\ldots(1-G_l)$$

$$= 1 - 1 + G_1 + G_2 + \ldots + G_l + O(G^2)$$

where $O(G^2)$ represents those terms involving the product of two or more G_μ. In general, however, grades of service are reasonably small and neglecting these terms allows the approximation

$$E = \sum_{\mu=1}^{l} G_\mu$$

As should be expected, the end to end GOS is always greater than each of the individual link grades of service. This probability of failure increases if the grades of service of the switches in the path are considered.

4.1.1.3. Degree of Rerouteing

In any network, some nodes will be connected directly together. Traffic between these two nodes will be carried by the link between them (assuming that this link is operable - alternate routeing should be provided to guard against failures). The primary routeing plan (the system for directing traffic around the network) is trivial in such a case.

Consider, however, two nodes i and j which, in contrast, are not connected directly together. The traffic a_{ij} must be rerouted via at least one other node. It is normal, in practice, to place an upper bound on the number of nodes in any such path to ensure satisfactory performance of the network. This bound is referred to as the degree of rerouteing (DOR). If every node is connected to every other node, the DOR is zero.

As with GOS, the DOR is part of the input to any NOP. Values of not more than 3 to 5 are common.

4.1.1.4. Cost of Equipment

The eventual cost of a telecommunications network is generally a combination of installation costs, connection charges and annual rental. The precise details of this vary from one application to another and with different times and conditions (and are often subject to considerable security (Upton [1985])). In some cases the projected development and expansion of the network as well as maintenance or repair work will be relevant. It may be that economic tools such as present value of annual charges (PVAC) or discounted cash flow (DCF) are required (Wherry [1975]). These are discounting techniques which allow a reduced sum of money to be invested to cover annual rental over a number of future years. Such methods are not considered in any detail here.

The actual costs of network components can be split into two general types; transmission related costs and switching related costs.

4.1.1.4.1. Transmission Costs

In general, L types of transmission circuit will be available. For each circuit type l, l=1,2,..,L, the cost C_l is often based on a somewhat involved set of rules (sometimes with the first circuit being markedly more expensive than subsequent circuits, etc. (Upton [1985])). These rules are intended to model the various ways in which costs are incurred in practice (such as by choosing certain types of cable and providing ducts of different sizes, etc.). A convenient approximation which may sometimes be used and which gives an idea of the form which transmission costs take is

$$C_l \;=\; F_l+I_l d$$

where F_l is a fixed cost and I_l is an incremental charge per unit length of circuit. d is the length of the circuit.

In practice, the method for determining the cost of a circuit may involve a step function with a different formula being used for two similar circuits of different length. The above approximation then becomes

$$C_l(d) \;=\; F_{l\sigma}+I_{l\sigma}d$$

where $F_{l\sigma}$ and $I_{l\sigma}$ are the appropriate multipliers for step σ. If D_σ and $D_{\sigma+1}$ are the lower and upper distance limits of the σ^{th} step, then σ is chosen so that $D_\sigma \le d \le D_{\sigma+1}$. Usually $F_{l\sigma}<F_{l\sigma+1}$ and $I_{l\sigma}>I_{l\sigma+1}$ (i.e. as the length of circuit increases, the fixed cost increases and the cost per unit length decreases).

One would expect the costs to be of this form. The provision of a transmission link will incur initial expenditure due to the cost of planning, supply of installation equipment, employment of workforce, etc.. In addition, the amount of cable provided and the work involved in laying it will increase with distance. Regular maintenance costs will also have their constant and distance dependent elements.

In general, these costs are passed on to the user in a similar form. Most will pay a fixed rental for equipment available on site and

Optimisation Techniques for Telecommunication Networks

V.M. Grout

Abstract

This thesis deals with various facets of the optimisation problem for telecommunication networks and proposes a number of new techniques for their solution.

The necessary essentials, Graph Theory, Complexity Theory and Telecommunication Principles, are investigated. The relevant graphs are enumerated and the requirements of suitable optimisation algorithms for certain graphical problems are established. The Private Automatic Branch Exchange (PABX) is introduced. The variety of telecommunications traffic as well as the practical requirements of a connection topology are discussed.

The fundamental Network Optimisation Problem (NOP) is defined and analysed. Simple exhaustive methods of solution are considered together with partial solution algorithms and simplification methods. Centralised networks with and without concentrators are introduced. Extensions and modifications are proposed for some techniques and existing practical methods of dealing with the NOP are investigated.

A number of new ideas are proposed for the practical solution of the NOP. Reduction methods are presented for replacing large unmanageable networks with smaller ones, on which optimisation can take place. Fixed topology techniques are introduced for initial tandem switch selection purposes and perturbation methods are considered which can be applied to such an initial solution. Lookahead methods of link removal are introduced for the purposes of determining the tandem interconnection network together with the traffic routeing strategy. A composite method is proposed incorporating all of these concepts and the results of a number of numerical experiments upon actual network problems are presented.

The extension of the proposed techniques to other areas of problem solving and optimisation is considered. In particular, a new method for the solution of the Euclidean Travelling Salesman Problem (ETSP) is presented.

A brief discussion is undertaken, in conclusion, concerning the practical difficulties of the NOP and the restrictions thus placed upon solution algorithms of various types.

call charges which are dependent upon the duration and distance of each call. In addition, an initial charge may be levied for the provision of facilities at the beginning of the contract.

4.1.1.4.2. Switching Costs

A <u>switch</u>, in practice, performs a relatively complex and important task within the network. For costing purposes, however, it may be regarded, in theoretical terms, simply as the termination of a number of circuits. Each circuit terminates at an individual <u>port</u>. The more circuits, the more ports, and the more ports, the more switching equipment required. The cost of a switch can therefore be represented by

$$C_s = F+Ip$$

where F is a fixed cost and I is an incremental charge per port. p is the number of ports.

As with transmission costs, this may be a simplification of the practical method by which switch costs are determined. In general, however, there is little deviation from this form (Upton [1985]). Again, step costs functions can be calculated as

$$C_s(p) = F_\sigma+I_\sigma p$$

where, this time, the steps are relative to the number of ports at the switch (i.e. $N_\sigma \le p \le N_{\sigma+1}$ where N_σ and $N_{\sigma+1}$ are the lower and upper limits (in terms of ports) of step σ.

Again, these costs are a combination of equipment expenditure, installation costs and regular maintenance. Ideally, these charges will be passed onto the user in a similar form as before.

The overall cost of any network will be the sum of the costs of its individual components.

$$C = \sum_{\mathfrak{S}} c_s + \sum_{l=1}^{L} \sum_{\mathfrak{L}_l} c_l$$

where $\mathfrak{S}$ is the set of all switches in the network
and $\mathfrak{L}_l$ is the set of all links of type l in the network.

4.1.2. Network Solution Requirements

The output from any NOP will be the full details of a completely connected network, of minimum cost, which satisfies the structure restrictions and the performance constraints. Necessary information will include those nodes which are connected together, the size of the connection between each pair of nodes, the amount of traffic flowing between each pair of nodes and the traffic routeing strategy as well as information regarding the management of the network. Fuller details are contained in section 4.3 and the next two chapters.

4.2. The Private Automatic Branch Exchange

The ultimate objective of any telecommunications network is to connect together a number of sources and sinks of traffic. Within this framework there are, however, two distinct requirements. Firstly, the sources and sinks within a certain area (building, site, etc.) need to be able to communicate. Secondly, access must be provided to other such areas of the country.

A <u>private automatic branch exchange</u> (PABX) represents the most powerful and effective way of achieving these objectives. Much of the work contained in the following chapters will be concerned with connecting together a number of distributed PABXs.

4.2.1. The Role of PABX

In the most general sense a PABX acts as a central point for the internal communications switching for a particular company in a given area. This area could typically be that covered by a firm's premises (or a regional centre for a national company). Sources and sinks of traffic are connected directly to the PABX to form a star (the simplest form possible).

The PABX may also provide a common point of connection to external public and private circuits for long distance communication. The <u>public switched telephone network</u> (PSTN), as an example, provides a complete nationwide structure for speech communication purposes. A number of other services, provided externally, such as <u>packet switchstream</u> (PPS, the UK national packet switched data network), can also be made available to the company user via the PABX.

A typical PABX structure is shown in Fig 4.b. Most will support both existing telephone equipment and more advanced terminals. Enhanced user options such as <u>automatic call diversion</u>, <u>abbreviated dialling</u> and <u>direct dialling in</u> are also provided. These are a number of operational facilities designed to give a more powerful and convenient service to the user. This generally uses the principle of <u>stored program control</u> (SPC) in which software is used to control the operation of the PABX. It allows for easy modification of, or addition to, the set of operating rules since only the relevant software need be changed to achieve any desired modification. A range of exchange sizes are possible up to several thousand extensions.

The common control equipment is often duplicated for security and reliability reasons. If one such piece of equipment fails, the duplicate can take over. This, combined with self testing diagnostic software - allowing potential and actual problems to be detected quickly, provides relatively fault free operation.

If necessary, a sophisticated console can be provided for use by a receptionist providing a manual operator system, if needed for any reason (such as in a situation where personnel move around frequently causing directories to become obsolete quickly), as well as an enquiry service. In addition, a management terminal may be available for call logging, analysis, route restriction, flow optimisation and general system monitoring and control.

Other facilities which may be provided by a typical PABX include personal computing (connection to a host computer simulating the effect of a desk-top PC), facsimile (a form of long distance

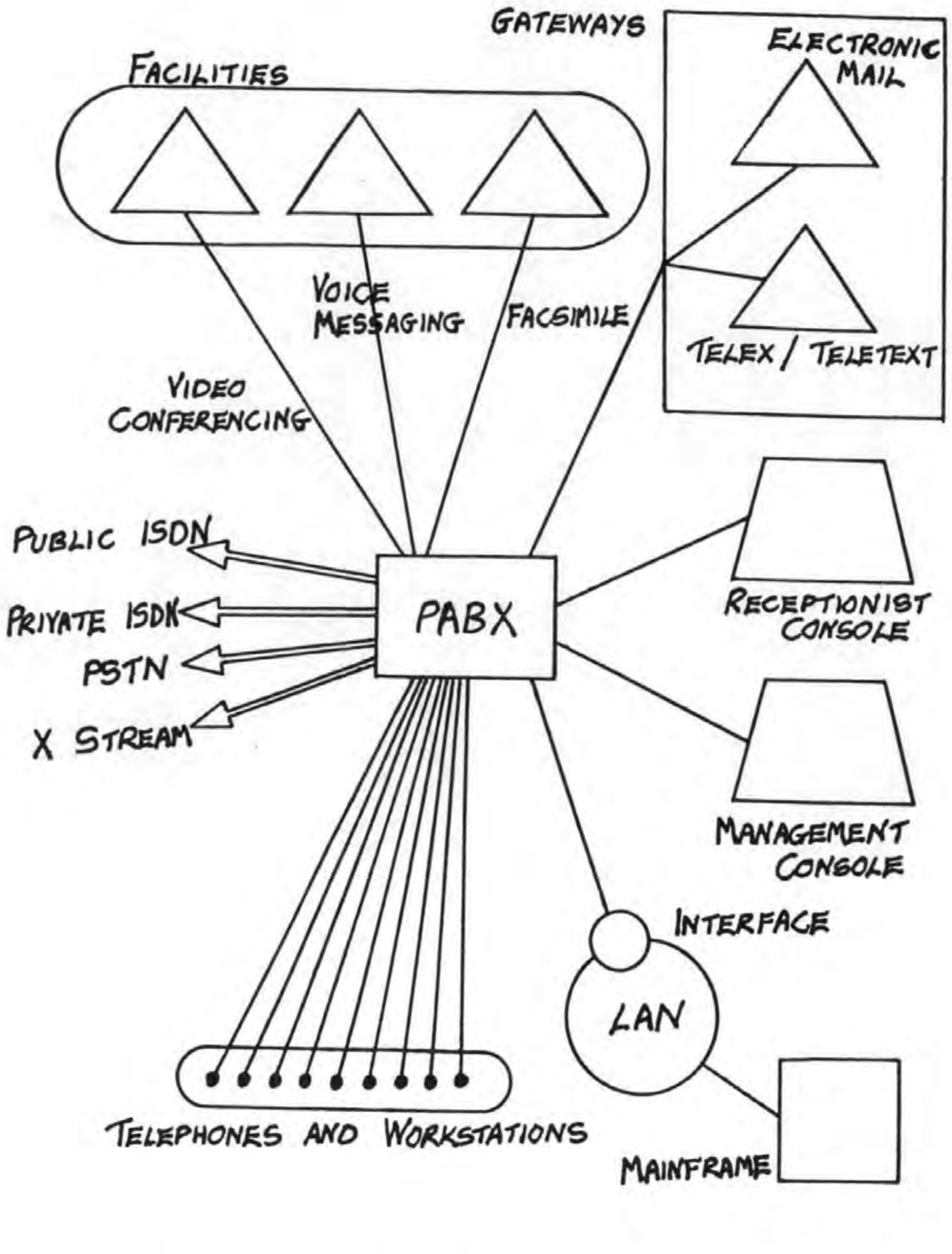


Figure 4.b. A Private Automatic Branch Exchange

photocopying in which any document, textual or graphical, can be sent over the network), video conferencing (the provision of multi-user conferencing facilities for seperated parties), voice messaging (a comparable system to electronic mail with audible messages as opposed to text), access to a local area network (LAN, an interconnected set of compatible terminals or stations), the integrated services digital network (ISDN, section 4.2.2) and gateways to electronic mail (a host computer simulating a number of mailboxes for its users and a 'postal' service between them) and telex and Teletext (messaging services between word processors and intelligent printers).

4.2.2. PABX and the Future

ISDN is very much the network of the future. Such systems will eventually unify the transmission of voice and data within the same structure.

The purpose of the ISDN is to provide a single pair of wires to each source/sink allowing for both digital voice and data to be sent and received. This includes all types of data discussed in the previous section. It will take some years before a full ISDN is established in the UK; but a pilot scheme exits at present and all new exchanges do have ISDN facilities. Future PABXs will be capable of full digital interworking with the ISDN.

Eventually, the ISDN will provide integrated digital access (IDA) to a range of network services (many of which are described in the previous section - others can be found in Lane [1984] and Gurrie & O'Connor [1986]) and will be capable of dealing with a number of different types of telecommunications traffic (voice, data, text, etc.).

There is, at present in the UK, an operational pilot scheme consisting of a number of ISDN local exchanges. Each user is connected to the appropriate local exchange via an 80kbit s^{-1} digital link. Alternatively, the service is available to PABXs via 2.048Mbits^{-1} PCM circuits (30 x 64kbit s^{-1}).

4.2.2.1. Centrex

A likely development in the next few years will be the continued introduction of <u>Centrex</u> or <u>business group services</u> (BGS). Centrex is a system allowing a local exchange to provide almost the same facilities that would be available from an advanced PABX. These facilities, however, are charged for on a rental and usage basis in much the same way as with existing phone facilities. For a company sending and receiving small amounts of data, the considerable expense of a PABX may not be justified. In these cases, the cost of subscribing to a Centrex local exchange will be preferable.

There are other reasons for which a company may choose Centrex to fulfill its PABX needs. There is no large initial outlay of capital in such a case and any subsequent improvements in technology and/or software can be provided without the need to reconfigure a PABX. In addition, the maintenance and operation of the system are likely to be superior since they are undertaken by the Centrex supplier. For example, all facilities should be available 24 hours per day - an unprofitable enterprise for a company running its own PABX.

Equally, however, there are disadvantages to such a system. At present, in the UK, only Mercury Communications provide an operational Centrex service and, even then, only in London. In addition, in this particular case, there is a minimum charge of 200 lines irrespective of actual company requirements. Furthermore, it is often necessary to duplicate equipment (such as multiplexers) and provide extra capacity to match the grade of service and level of reliability provided by a comparable PABX.

It is conceivable, in some cases, that a company could use a mixture of PABXs and Centrex exchanges to fulfill its transmission purposes at least cost. The costs are such, however, to ensure that large traffic generating sites are still best served by a separate PABX. This protects the local exchange from excessive amounts of traffic passing through it from a single source. The problem of PABX interconnection at minimum cost, to be discussed in chapter 5, consequently remains of importance.

4.3. PABX Network Principles

In isolation, a PABX provides a powerful solution to the local communication and service requirements of a small company or area group of a larger organisation. In the case of the latter, a number of PABXs will exist throughout a given area (the country, for example). It will generally be necessary to connect together these PABXs to form a complete network. This work is concerned with networks of a hierarchical form.

4.3.1. Hierarchical Network Structures

A number of PABXs (n say) can be considered as the unconnected nodes of the null graph N_n. It is thus required to find a suitable connected graph on these n nodes. The most obvious of these is K_n, the complete graph on n nodes. An alternative would be any of the n^{n-2} spanning trees (see chapter 2).

In practice, however, networks of the form shown in Fig 4.c are used. In this case, a subset of size M of the n PABXs have been selected and promoted to the position of <u>tandem PABXs</u>, or simply <u>tandems</u> (Lane [1984] and Gurrie & O'Connor [1986]). These tandems are connected in an unrestricted way. All other PABXs are connected directly to their closest tandem PABX.

It can be seen that these networks are, in fact, equivalent to the hierarchical nets discussed in chapter 2. There are therefore $({}^{n}C_{M})C_{M}$ acceptable networks with M tandems on n PABXs and

$$\sum_{M=1}^{n} \binom{n}{M} C_{M}$$

networks overall. The next two chapters discuss how to determine the cheapest such network (the optimum).

Two PABXs within the same tandem group communicate via their common tandem PABX. Traffic from a PABX to one in another group passes up into the <u>tandem network</u>, along a suitable route to the appropriate

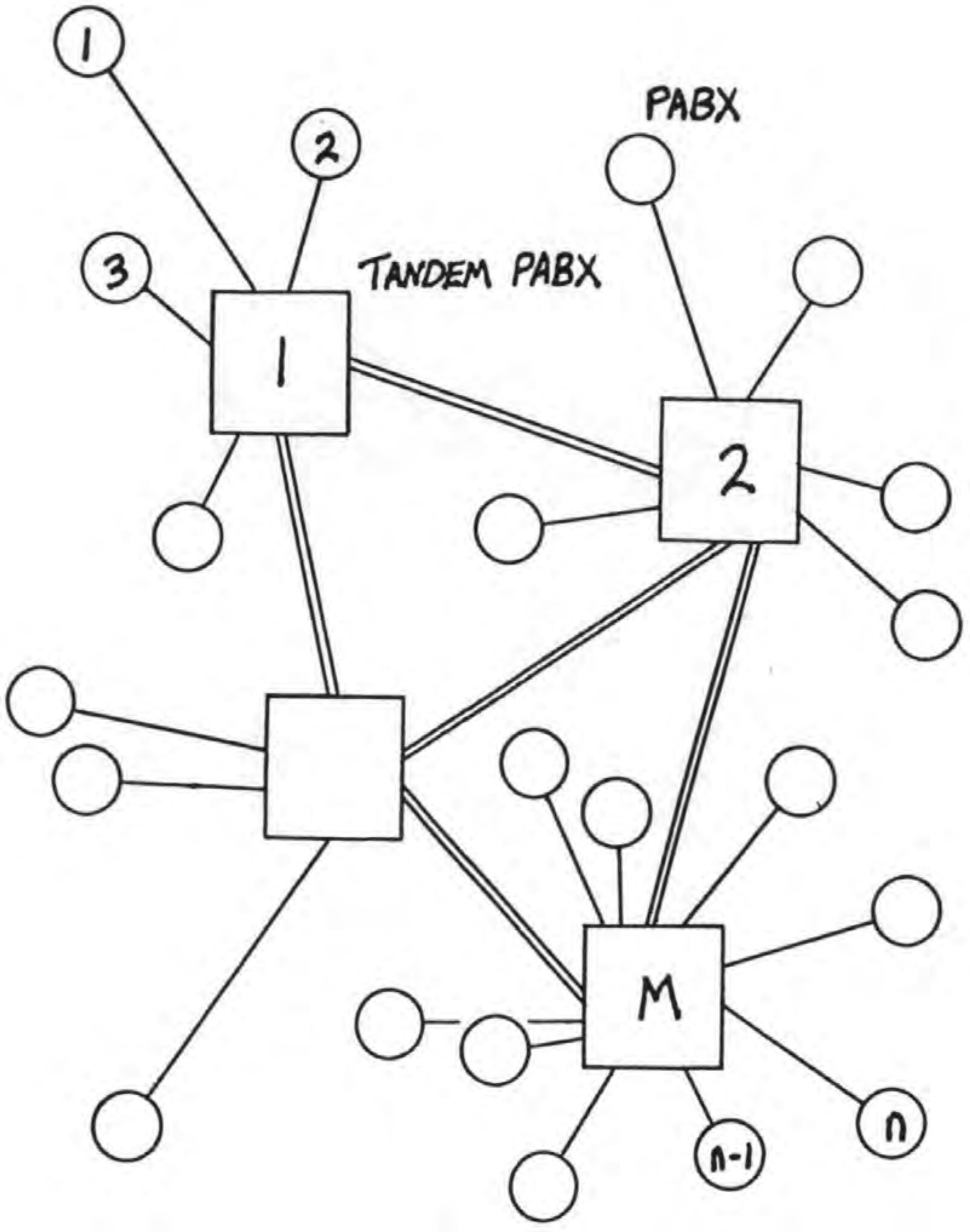


Figure 4.c. Typical PABX Network Structure

tandem and then down, through the <u>service</u> or <u>supply network</u>, to the target PABX. In this way the tandems collect and distribute traffic around the network.

This structure is particularly effective. A large proportion of traffic generated within a PABX is likely to remain local to that exchange or to one of its neighbouring exchanges. The multiple star arrangement at the lower level of the network (and within the PABX itself) deals with this well. The complete flexibility of the higher level, on the other hand, allows the tandems to be connected in the manner most appropriate to the characteristics and distribution of the traffic flowing between them.

4.3.2. Alternate Routeing Requirements

The arrangement described in section 4.3.1 is a simple and effective one and a reasonable choice for a practical PABX network (Lane [1984]). Certain situations, however, may require additional constraints on the network topology.

Consider two nodes (PABXs) i and j. The communications path between them is vulnerable in two specific places, namely the links from i and j to their parent tandems. If either of these links fail, i and j will become disconnected.

This potential problem can be avoided, to whatever extent is necessary, by the provision of alternate routeing in which every node is connected to P parent tandems where P>1. A primary (or preferred) route is given but if any of the links in this route fail the second, third, etc. routes can be brought into operation.

As far as the design (optimisation) of telecommunications networks is concerned, this type of alternate routeing is not difficult to provide. In the simple case with no such provision (i.e. P=1), each node is connected to its nearest tandem. To extend the case to a situation with alternate routeing of degree P is simple. Each node is merely connected to its nearest P tandems.

There are other forms of alternate routeing, however, which are considerably more difficult to deal with. A principal example is that of providing a given number of edge or node disjoint paths between each pair of nodes. In addition, nodes may, under certain circumstances, be connected to each other as well as to their parent tandem(s). In general, such problems are not considered in the work that is to follow. Each node will be assumed to be connected to a single tandem. Simple alternate routeing of degree P can then be achieved by the selection of P such tandems.

CHAPTER 5

TELECOMMUNICATION NETWORK OPTIMISATION

When a lot of remedies are suggested for a disease, that means it can't be cured

Anton Chekhov

The Cherry Orchard, 1904

Telecommunications networks, of any sort, are examples of practical engineering design. The variation among them is considerable. The larger structures are capable of carrying huge amounts of information, often over long distances. Such systems are unavoidably expensive due to the increased hardware needed for these purposes.

Clearly, the desire that a telecommunications network be designed to be as cheap as possible (in terms of whatever method the eventual cost is calculated), subject to constraints concerning the minimum level of acceptable performance, is an entirely reasonable one. A saving even of 1% for a network costing £10M is £100K. This easily justifies a considerable amount of effort at the design stage of the network project.

Unfortunately, however, such networks, by virtue of their practical nature, do not succumb to the principles of mathematical modelling very easily. Equipment costs are rarely calculated in a linear fashion, constraints may often take forms which prohibit their use within a model and many parameters required at the beginning of any simple process are often not known until the solution network is obtained.

This chapter is concerned with the problem of telecommunication network optimisation in its most general form. This problem is firstly stated and then analysed from a complexity point of view. In the subsequent sections a number of possible methods of solution, both for the complete problem and sections of it, are presented and considered.

The relative strengths and weaknesses of each are discussed along with the possibilities of extending some of the techniques to a more useful and practical set of circumstances.

5.1. The Problem of Network Optimisation

This section is concerned with the actual problem of determining the cheapest telecommunications network, in any particular case, which carries the appropriate quantities of traffic between all sources and sinks, at an acceptable level of service and subject to any given constraints. Firstly, a complete statement of the problem is given followed by a discussion of the relative difficulties involved in the solution of each of the resultant subproblems.

5.1.1. A Statement of the Problem

Much of the work that is to follow can be considered as being primarily concerned with the analysis and design of PABX networks. In the majority of cases, however, this distinction is academic in the sense that the techniques considered apply to any system involving sources and sinks of traffic. For this reason, any such traffic source/sink will be referred to, with reference to the graph theoretic case, as a <u>node</u> and the tandem PABXs etc. will be simply called <u>tandems</u> or <u>tandem nodes</u>. In those cases in which a particular type of network is being considered, the difference will be made clear

It is now possible to state the general <u>network optimisation problem</u> (NOP), defined by I1-5 and O1-10 of sections 5.1.1.1 and 5.1.1.2.

5.1.1.1. Required Input Data

There are two basic parts to the input for any NOP. The initial configuration of a network consists of node locations and traffic details. In addition, extra information is required to specify the environment in which the network is to operate. Clearly, the same network configuration could be presented as input to an NOP with a variety of different environments.

The required input to any NOP can be explicitly defined as follows.

I1. The coordinates (x_i, y_i), $i=1,2,..,n$ of n nodes in the 2-D Euclidean plane.

I2. The values contained in T traffic matrices $A_\tau = (a_{\tau ij})$, $i,j=1,2,..,n$, $\tau=1,2,..,T$ where there are T types of traffic and $a_{\tau ij}$ represents the amount of traffic of type τ, in erlangs, originating at source i and destined for sink j.

I3. The GOS required on any link within the final network. This GOS may be different for each level of the solution network.

I4. The hardware available for use in the solution and the formulae or strategies to be used in determining the cost of each piece of equipment.

I5. Additional, optional constraints on the final solution such as DOR or level of connectedness (LOC - the minimum degree of each tandem node), fixed links and/or tandems, etc..

The appropriate form of each element of input will be discussed in section 5.1.2.1.

5.1.1.2. Required Output Data

The following output will completely describe the solution to a general NOP. It should be either given by, or easily obtainable from, any calculated solution.

O1. The optimum number of tandem nodes, Ω, $1 \le \Omega \le n$.

O2 The optimum selection of Ω tandem nodes from the original n.

O3. The optimum partitioning of the remaining n-Ω nodes into the Ω tandem groups.

O4. The most suitable link type for each node-tandem connection.

O5. The required size of each node-tandem connection link.

O6. The optimum tandem network connection topology.

O7. The most suitable link type for each tandem-tandem connection.

O8. The required size of each tandem-tandem connection link.

O9. The optimum traffic routeing strategy within the tandem network.

O10. Numerical information concerning the solution network such as the cost of the network, individual grades of service (under conditions of correct operation and component failure), reliability, etc..

Some of these individual solutions are easier to obtain than others. Section 5.1.2.2 considers the relative problems involved in each case.

5.1.2. Analysis of the Problem

The general NOP is clearly easy to state. It will be seen, however that its solution is far more difficult. This section begins by explaining the input data in greater detail and then proceeds with an investigation of the problems encountered in determining each element of the output.

5.1.2.1. Analysis of Input Data

The node locations (I1) can be described by any valid set of 2-D coordinates relative to any Euclidean frame of reference. In the UK, the national Ordinance Survey grid references of 0-700km (west-east), 0-1000km (south-north) are typically used with as much definition as required. Latitude and Longitude may be used if wished but, in general, it will be necessary to approximate them by coordinates on a flat surface.

The traffic matrices $A_\tau = (a_{\tau ij})$, $i,j=1,2,\dots,n$, $\tau=1,2,\dots,T$ (I2) are given separately. In some cases, it may be possible to combine traffic or even to represent traffic between two nodes as a single figure, giving rise to a single matrix A. The amount of traffic together with the required GOS (I3) will determine the necessary size of each link. The form of the equation in section 4.1.1.2 implies that if $C(G,A)$ is the number of circuits required to carry A erlangs at GOS G, then $C(G,A_1+A_2) \le C(G,A_1)+C(G,A_2)$. Combination of traffic is therefore desirable wherever possible. Typical grades of service may range between 0.001 (0.1%) and 0.1 (10%). Specifying link grades of service ensures that end to end grades of service are better for shorter routes than long ones, even if there is a different GOS for the tandem network and the remainder.

The available hardware may vary quite considerably. Also the costing strategies may not be trivial (I4). It will, however, generally be possible to evaluate the cost of equipment (and hence of entire networks) in whatever way it is achieved in the real world since only a finite variety of equipment can be available and some method must exist for calculating the price of each.

The use of the optional constraints (I5) will vary from application to application. DOR is the most common, and indeed is nearly always used. Other constraints may be implemented whenever necessary. Examples of these optional constraints include

C1. Insisting that each tandem is connected to at least x other tandems (LOC).

C2. Insisting that a certain node act as a tandem.

C3. Insisting that a certain node not act as a tandem.

C4. Insisting upon the existence of a certain link within the tandem network.

C5. Insisting that a certain link not exist within the tandem network.

C6. Insisting that a certain link carry no rerouted (i.e. extra) traffic within the core network.

Clearly, any constraints will serve to decrease the number of acceptable solution networks. In principle at least then, the corresponding NOP should be made simpler. In practice, however, it is often necessary to generate a solution before its acceptability can be tested, thus negating this effect. This, in conjunction with the appropriate new testing requirements, usually mean that additional constraints actually complicate the problem and increase solution time.

There are some cases where constraints can be of assistance though. These are dealt with later.

One particular form of constraint needs to be considered further. It is often the case that, in addition to a number of erlangs of traffic flowing from one node to another, a number of fixed circuits are required as well. Traffic carried on these circuits is often referred to as <u>non-switched traffic</u> as opposed to the <u>switched traffic</u> more regularly encountered. It is so called because, despite being routed via a number of intermediate nodes en route from source to sink, no actual switching takes place. Unlike the switched traffic matrices, the non-switched traffic matrix must be symmetric since the number of non-switched circuits will be the same between two points in either direction.

If the total switched traffic on the link from i to j is given by s_{ij} and from j to i by s_{ji} and f_{ij} fixed non-switched circuits are required, then the total number of circuits required on the link will be given by $C = f_{ij} + E(s_{ij}+s_{ji})$ where $E(x)$ represents the number of circuits required to carry x erlangs of traffic at the current GOS. Non-switched traffic then presents no particular problem and will not be discussed explicitly again.

5.1.2.2. Analysis of Output Data

Clearly, the optimum number of tandem nodes Ω (O1), lies between 1 and the number of nodes. Its value will depend primarily on the amount of traffic which the solution network has to carry and the relationship between transmission and switching costs. The number of acceptable solution networks on n nodes is therefore of the following form.

$$N_n = \sum_{M=1}^{n} f(n,M)$$

N_n can be thought of as the number of networks which an exhaustive search algorithm would have to generate and test in order to find the cheapest solution. $f(n,M)$ is the number of acceptable solution networks with M tandems.

The optimum selection of Ω tandem nodes from n (O2) can be generalised to finding the optimum selection of M tandem nodes, where M is the number of tandems being tested at any one time. Therefore

$$N_n = \sum_{M=1}^{n} \binom{n}{M} f'(n,M)$$

where $f'(n,M)$ is the number of valid solutions with a fixed choice of M tandems.

The optimum partitioning, or grouping, of the nodes into the tandem groups (O3) can be considered in two ways. Firstly, there are n-M ungrouped nodes when there are M tandems with M possible groups for each. There are therefore M^{n-M} possible groupings in all and

$$N_n = \sum_{M=1}^{n} \binom{n}{M} M^{n-M} f''(n,M)$$

where f"(n,M) is the number of valid solutions with a fixed choice of M tandems and a particular node partition structure.

A common, and not unrealistic assumption, however, is that each node is to be attached to its nearest tandem (or tandems if alternate routeing is provided) in the plane. In general, this will minimise cost irrespective of the topology of the rest of the network. There are occasions when it may be profitable to connect a node to a distant tandem in order to avoid overloading the nearest but, for most practical purposes, this simplification is adopted, as it will be here. Hence the number of acceptable networks reverts to

$$N_n = \sum_{M=1}^{n} \binom{n}{M} f''(n,M)$$

There will be a number of alternative choices of transmission medium in most cases. If the location of the tandem nodes is established, then the amount of traffic flowing between each node and its nearest tandem, as well as the distance it travels, is known. The most appropriate piece of hardware for each node-tandem link (O4) can be found by simply evaluating the cost of each and choosing the cheapest. The cost of each link, however, will depend on its size (O5). This can be determined from the tables in appendix B since it will be a function of the amount of traffic it must carry and the GOS which must be provided. In practical terms, these calculations do not add to the complexity of the problem.

Determining the connection topology of the tandem network (O6), however, does form a major part of the complete problem. The number of possible solutions, in an unconstrained case, is equal to the number of connected graphs on M tandems, which is bounded below by $2^{(M(M-1)/2)-1}$ (chapter 2). This, in turn, implies that

$$N_n \geq \sum_{M=1}^{n} \binom{n}{M} 2^{(M(M-1)/2)-1} f'''(n,M)$$

The most suitable link type for each tandem-tandem connection (O7) and its required size (O8) will be determined by the amount of traffic it has to carry. This will depend, not only upon the physical topology but also, upon the traffic routeing strategy. Once this routeing is known, the type and size of each link can be chosen and calculated respectively in the same way as with the node-tandem links (O4 & O5).

The problem of determining the optimum traffic routeing strategy, however, (O9) is a potentially massive task and may be the single most difficult section of the complete optimisation process. For any two tandems, not directly connected, their common traffic can be rerouted via up to D-2 other tandems where D is the DOR (if defined - otherwise D=M). These D-2 tandems can be chosen in ${}^{M-2}C_{D-2}$ ways and the problem is repeated for each of the u pairs of nodes, not directly connected. Clearly u depends on the topology of the tandem network (O6) so an explicit form is not practical. The number of possibilities, however, is of the form $O(u^{(M-2)!/(D-2)!(M-D)!})$, showing clearly, the size of the problem in hand. If the number of possible routeing strategies for a given topology is R then the number of possible solution networks is given by

$$N_n \geq \sum_{M=1}^{n} \binom{n}{M} 2^{(M(M-1)/2)-1} R$$

The remaining information (O10) is not, as such, part of the optimisation process. The solution network has to be established before such data can be calculated. Consequently, no extra complexity is incurred. Any required information can merely be calculated directly from details of the solution network (Van Slyke & Frank [1972], Wilkov [1972] and Kleinrock [1979]).

Putting R=1 in the above equation provides a very loose lower bound on N_n.

$$L_n = \sum_{M=1}^{n} \binom{n}{M} 2^{(M(M-1)/2)-1}$$

The first few values of L_n are as follows (subsequent calculation becomes very difficult).

n	L_n	n	L_n	n	L_n	n	L_n
1	0.5	2	2	3	8.5	4	56
5	724.5	6	20034	7	10^{6}	8	10^{8}
9	10^{10}	10	10^{13}	11	10^{16}	12	10^{19}
13	10^{23}	14	10^{27}	15	10^{31}	16	10^{36}
17	10^{40}	18	10^{46}	19	10^{51}	20	10^{57}
21	10^{63}	22	10^{69}	23	10^{76}	24	10^{83}
25	10^{90}	26	10^{97}	27	10^{105}	28	10^{113}
29	10^{122}	30	10^{130}	31	10^{139}	32	10^{149}
33	10^{158}	34	10^{168}	35	10^{179}	36	10^{189}
37	10^{200}	38	10^{211}	39	10^{223}	40	10^{234}

In practice of course, R>>1 and may be, in fact, the dominant part of N_n. It is clearly infeasible, as the number of nodes increases, to implement an algorithm which simply generates and prices each solution network in turn. Nevertheless, such an approach does provide an excellent starting point for a general analysis of the problem and can be used for limited testing of results.

5.1.3. Finding the Cost of a Solution Network

Define a crude solution network to be a form of solution network in which outputs O1, O2, .., O8 and O9 are determined but O10 is not. The practical component of the problem has been solved and it only remains to calculate various pieces of information associated with the cost and performance of the resultant network. As suggested in section 5.1.2.2, this information does not present significant difficulties with respect to the complexity of the rest of the problem.

This section illustrates this point by considering the calculation of the cost of a crude solution network. It also provides justification of a certain aspect of the forthcoming exhaustive search algorithm.

There will be four basic contributions to the total cost of a two level hierarchical network.

CO1. The cost of the nodes.

CO2. The cost of the node-tandem links.

CO3. The cost of the tandem-tandem links.

CO4. The cost of the tandem nodes.

CO1 will remain constant. The node will generate and receive the same amount of traffic irrespective of the configuration of the rest of the network and thus possess the same number of ports. The tandem to which it is connected is irrelevant in this sense. It is customary, in many cases, to go one stage further. CO1 is eliminated from the cost of the network rather than carried through the process as an invariant. If, however, this figure is required, it can be calculated simply by determining all traffic which is incident at each node, finding the number of circuits required at each (and hence the number of ports) using the Erlang formula, applying the appropriate cost formula and summing over all nodes.

The location of all of the transits is established. Consequently, the distance from each node to its nearest transit is known also. In addition, the size and type of each node-tandem link is known. The appropriate cost formula can then be used in each case to calculate the cost of each link. Summing over all node-tandem links will give CO2.

In a similar way the distances between all pairs of tandems are known. It is furthermore known which pairs have a link between them and the size and type of these links. As is the case with the node-tandem links, the tandem-tandem links can be priced according to the appropriate formula. The sum of these costs over all links present within the tandem network will give CO3.

It is known which nodes have each tandem as parent and how many circuits are associated with this number of nodes. Therefore, the number of ports required on the service network side of each tandem is known also. It is furthermore known which of the others each tandem is connected to and how many circuits are required on each link. The number of ports on the tandem network side can thus be determined as well. This information can be substituted into the relevant formula to find the cost of each tandem. The sum of these costs give CO4.

The overall cost of any crude solution network can therefore be calculated as CO2+CO3+CO4 (+CO1 if necessary). It therefore presents no great difficulty to find the cost of any given network and the instruction

calculate cost of network N

within an algorithm need not be considered as anything more complex than a predefined procedure for the above purpose.

5.2. Exhaustive Search Techniques

This section sets out to produce an algorithm which will systematically test every feasible solution network. Throughout the algorithm, the cost of the cheapest network will be retained, along with the appropriate details of that network for eventual output. If, at any stage, a new network is encountered, satisfying the performance constraints, which is cheaper than the existing best, then the current network becomes the retained network.

Whilst it is impractical to use such an algorithm in any realistic sense, a modified version of it is used, with success, in chapter 7 in order to provide exact solutions against which the solutions from a heuristic (approximate) method can be compared.

5.2.1. The Exhaustive Search Algorithm

The form of any exhaustive search algorithm will most naturally follow the nested loop structure suggested implicitly by the analysis of section 5.1.2.2. The structure of the network can be considered as similar to a multidimensional array in which all indices are varied over their full range in order to cover all possibilities.

It is necessary to place a restriction, for the purposes of practicability, on the DOR of the solution network. Instead of defining a single DOR for the entire network, the constraint will be that the shortest path (in terms of the number of links) will always be chosen between two tandems. This includes, of course, the direct link between them if one exists for the particular topology in question. For some node pairs, paths of equal lengths will be in contention. In such cases, the path involving the shortest Euclidean distance will be chosen.

For the purposes of this algorithm and those that follow, abbreviated input and output parameter sets will be defined for convenience. Let the required input and output data of section 5.1.1 (I1-5 and O1-10) be represented by the sets $\mathcal{I}$ and $\mathcal{O}$ such that

```
I:=𝓘
:
algorithm
:
O:=𝓞
```

is equivalent to

$I:\{x_1,y_1,x_2,y_2,\ldots,x_i,y_j,\ldots,x_n,y_n,a_{111},a_{112},a_{113},\ldots,a_{\tau ij},\ldots$
$\ldots,a_{Tnn}$, GOS, cost formulae, DOR, LOC, ..., optional constraints$\}$

:

algorithm

:

$O:\{\Omega$, tandem choices, node-tandem link types, node-tandem link sizes, tandem network connection topology, tandem-tandem link types, tandem-tandem link sizes, routeing strategy, additional information (O10)}

The input parameter set can be considered as representing an initial network N_1, and the output parameter set, a solution network N_s. In addition, N represents the current network under consideration. This is a particularly useful general notation since not every element of input and/or output will be relevant in a given situation. A suitable algorithm can then be constructed as follows.

```
I:=I ;
minimum := ∞ ;
for M := 1 to n do
    BEGIN
    choose_first_set_of_tandems ;
    REPEAT BEGIN
           choose_first_tandem_network_connection_topology ;
           REPEAT BEGIN
                  determine_tandem_network_routeing_strategy ;
                  calculate_cost_of_network(N, cost) ;
                  IF cost<minimum THEN
                     BEGIN
                     N_s := N ;
                     minimum := cost ;
                     END ;
                  IF there_are_more_topologies_to_test THEN
                     generate_next_topology ;
                  END ;
                  UNTIL there_are_no_further_topologies_to_test ;
           IF_there_are_more_tandem_sets_to_test THEN
              generate_next_tandem_set ;
           END ;
           UNTIL there_are_no_further_tandem_sets_to_test ;
    END ;
O:=O .
```

The optimum network, on termination of this algorithm, will be the network N_s whilst the cost of N_s will be the variable, minimum.

5.2.2. Requirements of the Algorithm

The algorithm of section 5.2.1. does indeed provide a method for exactly finding the optimum network by exhaustive search. There are, however, some commands within it which are not, as yet, sufficiently explicit. In particular, this refers to the operations specified by the commands generate_next_topology and generate_next_tandem_set.

The next two sections discuss these commands. Note that the command to calculate the cost of a particular network is not a problem (section 5.1.3) and that because of the extremely restrictive DOR constraint, neither is the command to determine the tandem network routeing strategy.

5.2.2.1. Generating Tandem Sets

Within the exhaustive search algorithm of section 5.2.1 there is the following structure.

```
choose_first_set_of_tandems ;
REPEAT BEGIN
          :
          IF there_are_more_tandem_sets_to_test THEN
             generate_next_tandem_set ;
          END
          UNTIL there_are_no_further_tandem_sets_to_test ;
```

For this code to operate as suggested, three things need to be known:

1. The first set of tandems,
2. The way in which each set of tandems is selected in turn,
3. The stopping criterion.

The most obvious starting set of M tandems from n nodes is the first M nodes listed in order (i.e. nodes i=1,2,..,M). Each set of tandems could then be generated logically up to the final M nodes listed in order (i.e. i=n-M+1,n-M+2,..,n).

Appendix A contains a Pascal procedure to effect this transit set updating process without missing any tandem sets.

5.2.2.2. Generating Tandem Network Connection Topologies

The exhaustive search algorithm also contains the code for generating the tandem network connection topology.

```
choose_first_tandem_network_connection_topology ;
REPEAT BEGIN
          :
          IF there_are_more_topologies_to_test THEN
             generate_next_topology ;
          END
          UNTIL there_are_no_further_topologies_to_test ;
```

Again, there are three requirements - the starting and finishing topologies and the updating procedure. The most natural topology to begin with, despite it not being acceptable as a connected network, is the one in which no tandem is connected to any other (i.e. the null graph on M tandems). In this case, the obvious finishing topology is the complete graph on M tandems. At each stage only connected topologies are considered.

Appendix A contains an algorithm which will start and finish with these connection topologies and systematically generate each one in between.

5.2.3. Complexity of the Algorithm

An examination of the loop structure of the exhaustive search algorithm shows its computational complexity to be $O(2^{n^2})$. Comparing this with the O(n!) complexity of a similar exhaustive search algorithm for the TSP (Johnson & Papadimitriou [1985]) shows the magnitude of the task.

Let $f(n) = n!$.
Then $f(n+1) = (n+1)! = (n+1)n! = (n+1)f(n)$.

Let $g(n) = 2^{n^2}$.
Then $g(n+1) = 2^{(n+1)^2} = 2^{n^2+2n+1} = 2^{n^2}2^{2n+1} = (2^{2n+1})g(n)$.

The TSP is one of the classical NP-complete problems in combinatorial complexity theory. Clearly, some considerable improvement is required before an acceptable algorithm for the NOP, for large values of n, is achieved.

5.3. Simplifications and Special Cases

The unconstrained NOP is a task of massive proportions. It represents the most general case of all in the field of network design. There are occasions, however, when this generality is not required. Alternatively, it may be regarded as acceptable, in certain situations, to make some assumptions and simplifications in order to decrease the amount of effort expended in obtaining a solution. This leads to a number of ways in which the problem can be solved using techniques which would not normally be acceptable. This section considers two such cases. Both are products of some form of constant cost assumption.

5.3.1. Centralised Communication Networks

The general NOP deals with networks in which communication may be required between any two nodes i and j. In contrast, there are networks in which all nodes require communication with a central node only. Examples include airline inquiry/booking systems and bank service till networks.

In such systems, the dependent nodes are labelled $i=1,2,\ldots,n$ while the central node is labelled 0. The traffic is such that $a_{ij}>0$ only if $i=0$ or $j=0$.

Much of the work carried out in this area to date has been with respect to computer communications networks. The techniques can be extended, however, to provide a more general approach.

Each node i $(i>0)$ sends traffic a_{i0} to the central node and receives traffic a_{0i} from it. Let $a_i = a_{i0}+a_{0i}$ so that a_i represents the total amount of traffic flowing between the node i and the centre.

The cost of the network is to be determined in a simplified manner. Switching only takes place at the centre and this node will always have to handle the same amount of traffic whatever the configuration of the network. Disregarding small differences in port requirements allows the switch costs to be considered as constant and thus ignored in the optimisation process.

The link between any node pair is considered to be of fixed capacity and can have two states within the network - present or absent. If a link is present in a particular solution network, it will always cost the same amount, irrespective of the amount of traffic it carries. This traffic, however, has an upper limit, E. This allows the cost of the links to be represented by a matrix $C=(c_{ij})$ where c_{ij} gives the cost of the link from i to j and is usually primarily dependent upon distance.

The values of $\underline{a}$, C and E are the input to the simplified problem. This is to find the network which carries all the traffic from the nodes to the centre, at the minimum cost, and without exceeding the value of E on any individual link. A typical (but small) problem is shown in Fig 5.a. It is not necessary to define explicitly the units of cost or distance (they could be erlangs, ccs h^{-1}, km, etc. as required). The solutions will adopt a spanning tree topology of the form shown in Fig 5.b.

5.3.1.1. Exact Methods of Solution

With examples such as in Fig 5.a, of course, it is quite feasible to generate all solutions and choose the cheapest. As the number of

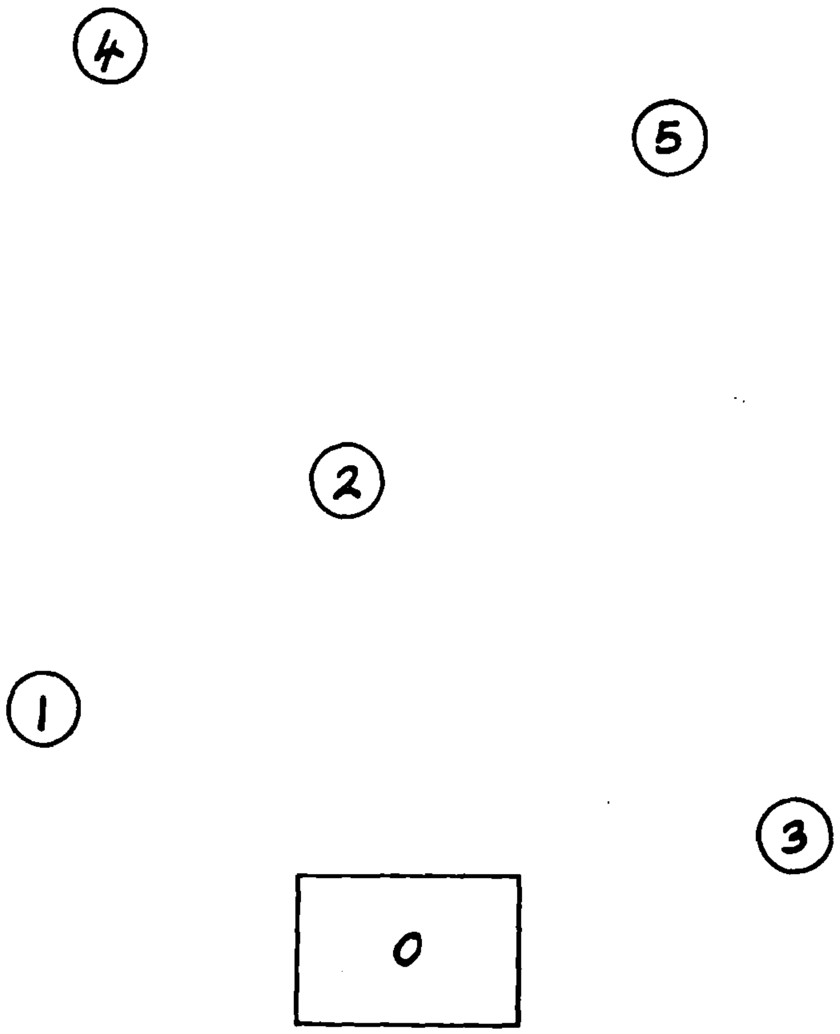


Figure 5.a. Centralised Network: Initial Configuration

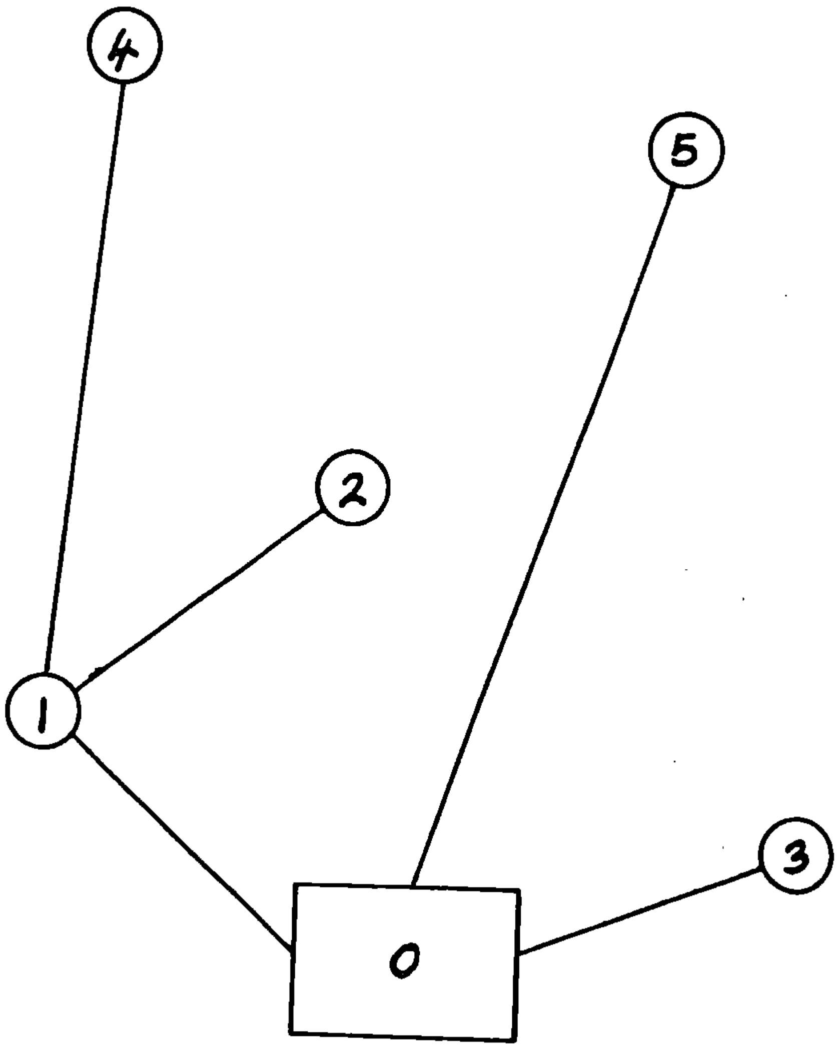


Figure 5.b. Centralised Network: A Possible Solution

nodes increases, however, the number of spanning trees (n^{n-2}) becomes much too large. In addition, the techniques needed for selecting each tree in turn become extremely complicated.

A slight improvement to this approach, in the sense that it provides a more logical and efficient tree generating strategy, is suggested by Chandy & Russell [1972] based on the earlier branch and bound work of Little et al. [1963]. The method works by repetitively partitioning the solution space into progressively smaller subspaces. At each stage, the subspace containing the optimum is determined and the solutions contained in the other subspace are discarded. Eventually, the optimum is reached. The actual algorithm is as follows. The term feasible is used to describe a solution in which the value of E is not exceeded.

CR1. Construct the minimal spanning tree on the n+1 nodes. If this solution is feasible then stop; Otherwise continue with CR2.

CR2. Select those nodes, $i_1, i_2, \ldots, i_m$ which are incident to node 0 in this solution. Form a solution subspace A, containing all solutions involving the links $(0,i_1), (0,i_2), \ldots, (0,i_m)$.

CR3. Select a link (i,j), not already chosen. Partition A into subspaces A' and A" where A' is the subspace of A containing solutions involving the link (i,j) and A" is the complement of A' in A.

CR4. Construct MST solutions for each subspace. Calculate the cost, C' and C", of each. If the smaller of these is equal to the cost of a feasible solution in the same subspace then the solution is found. Otherwise continue with step CR5.

CR5. If C'<C" then relabel A' as A. Otherwise relabel A" as A. Goto CR3.

Step CR2 is based on the fact, proven by Chandy & Russell [1972], that any direct link to the centre in the optimum unconstrained solution is also present in the constrained version.

As an example, consider the example of Fig 5.a with a_1:6, a_2:4, a_3:3, a_4:5, a_5:7, E:10 and costs given by

$$C \; : \; \begin{array}{c} \left(\begin{array}{cccccc} - & 5 & 3 & 6 & 9 & 10 \\ 5 & - & 4 & 11 & 6 & 9 \\ 3 & 4 & - & 7 & 5 & 7 \\ 6 & 11 & 7 & - & 8 & 5 \\ 9 & 6 & 5 & 8 & - & 6 \\ 10 & 9 & 7 & 5 & 6 & - \end{array} \right) \end{array} \qquad \begin{array}{c} i : 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}$$

The unconstrained MST is shown in Fig 5.c. This solution is infeasible since there are 15 units of traffic on the link (0,2). The subspace A consists of all those solutions involving the links (0,2) and (0,3). Construct subspaces A' and A" of A where all solutions in A' involve the link (2,4) (chosen arbitrarily) and no solution in A" contains (2,4). The constrained MSTs for each case are shown in Figs 5.d and 5.e. Their costs are 25 and 28 respectively. If necessary, A' would now be renamed A and the process repeated. However, the solution from A' is not only locally optimal, it is feasible as well. The optimum solution is thus the one shown in Fig 5.d at a cost of 25.

5.3.1.2. Heuristic Methods of Solution

The methods of exhaustive search and modified branch and bound, despite providing exact solutions, require too much time to be of practical use. Branch and bound techniques are potentially as complex as ordinary exhaustive search and neither method is at all satisfactory.

The obvious alternatives to these exact methods are techniques which run in a far shorter time, but which are not guaranteed to produce the optimum. So long as the results are generally within acceptable limits, these techniques may be of use. A number of such methods are presented in this section.

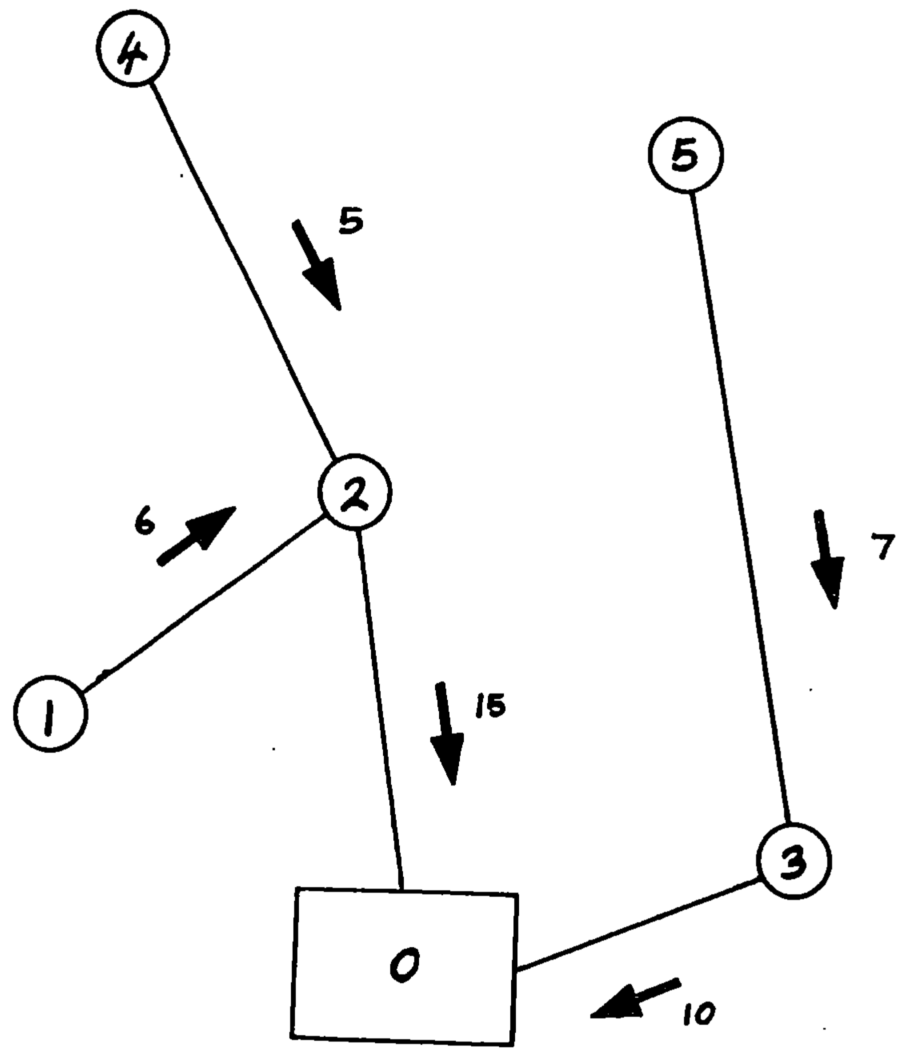


Figure 5.c. Centralised Network: Unconstrained MST

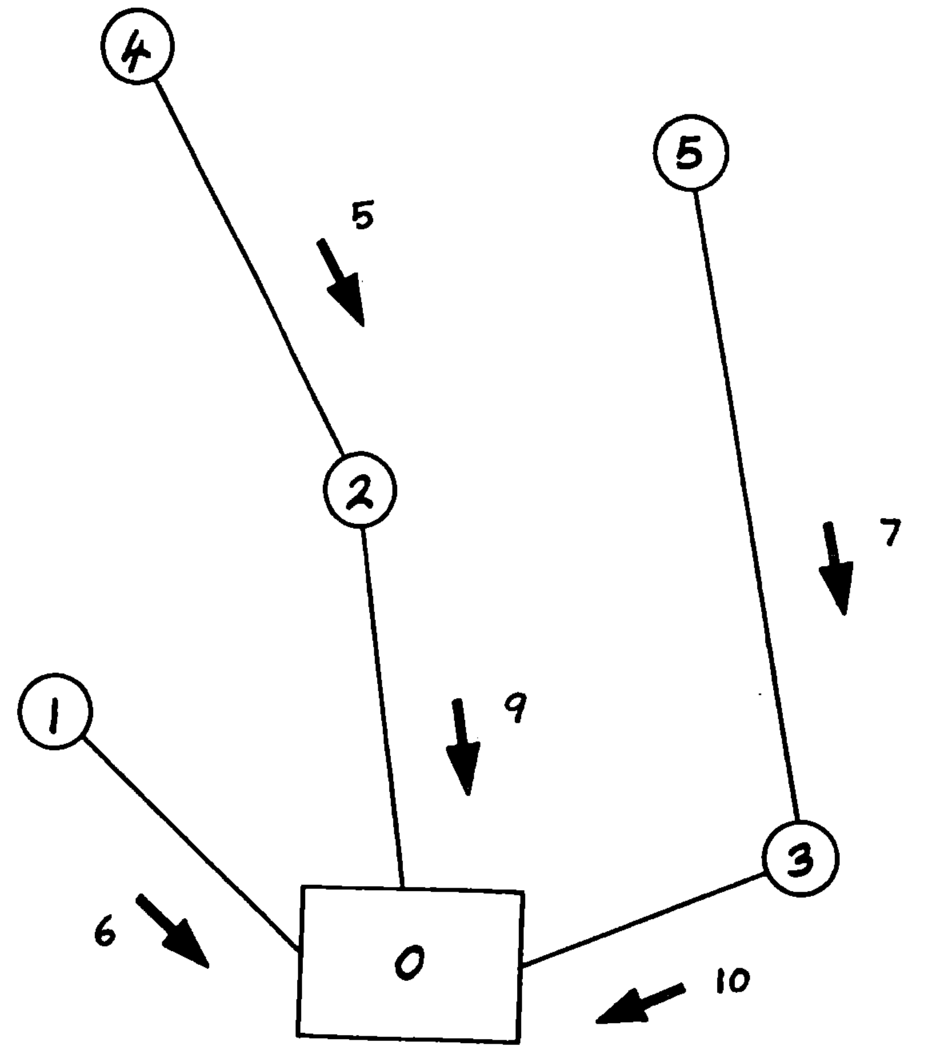


Figure 5.d Centralised Network: Constrained MST for Subspace A'

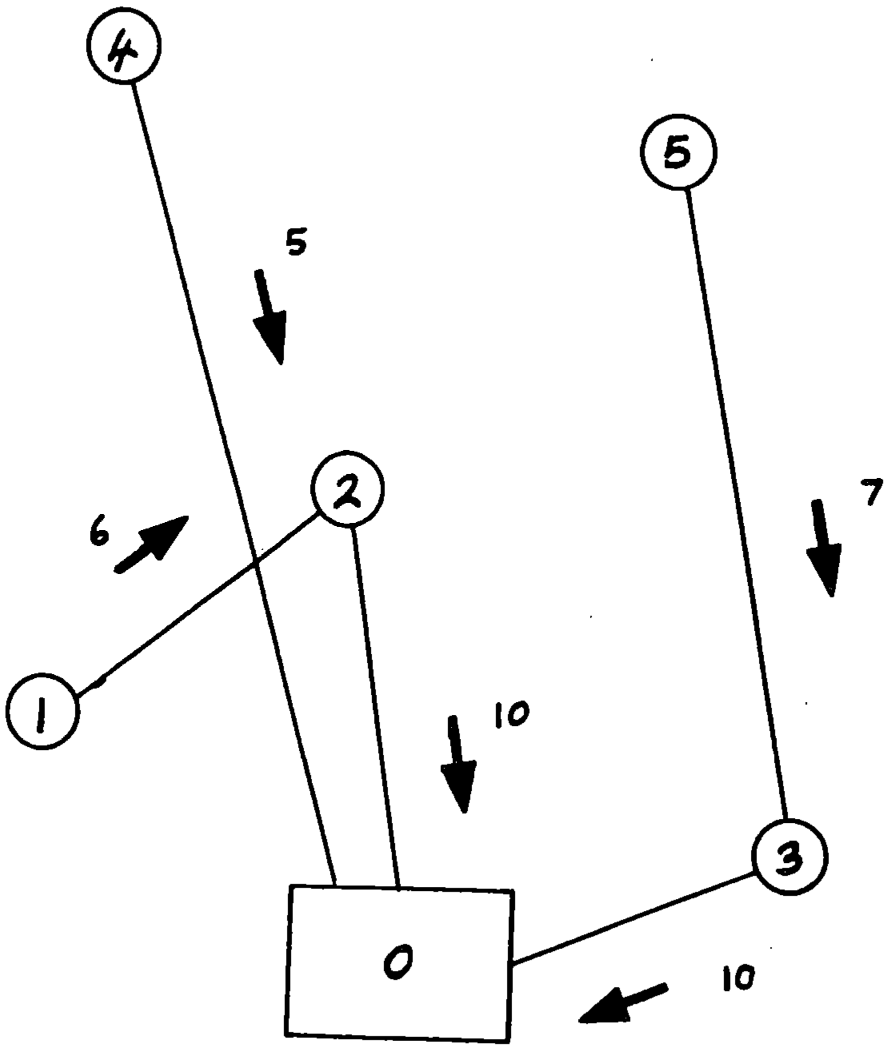


Figure 5.e. Centralised Network: Constrained MST for Subspace A"

5.3.1.2.1. The Constrained Kruskal Algorithm

This is a simple extension of the algorithm in section 2.3.1 (Kruskal [1956]). It chooses the cheapest link at each stage which does not violate the E constraint or form a circuit.

CK1. Construct a sequence of n unique links $l_1, l_2, .. l_n$, choosing each new link, l_i, as described in CK2.

CK2. l_i is the unchosen link of least cost c_i which forms no circuit with links $l_1, l_2, .., l_{i-1}$ and does not cause the traffic on any link to exceed E.

The solution obtained when using this method on the configuration of Fig 5.a is the same as that in Fig 5.e. The cost is 28, 3 more than the optimum.

5.3.1.2.2. The Constrained Prim Algorithm

This is only a slight variation of the constrained Kruskal algorithm. It is adapted from Prim [1957].

CP1. Construct a sequence of n unique links $l_1, l_2, .. l_n$, choosing each new link, l_i, as described in CP2 and CP3.

CP2. l_i is the unchosen link of least cost c_i which forms no circuit with links $l_1, l_2, .., l_{i-1}$ and does not cause the traffic on any link to exceed E.

CP3. Each link must connect a node to the tree spreading from the centre (- for example, in Fig 5.e, the link (1,2) cannot be added before one of the links (0,1) or (0,2)).

In the example of Fig 5.a, the solution obtained is the same as with Kruskal's algorithm.

5.3.1.2.3. The Esau-Williams Algorithm

The Esau-Williams algorithm (Esau & Williams [1966]) is a slight improvement on the above methods. It begins with all nodes connected to the centre and then systematically reroutes traffic according to the maximum improvement in cost.

EW1. Connect all nodes to the centre. Calculate $T_{ij}=c_{i0}-c_{ij}$ for each pair (i, j).

EW2. Find the pair (i',j') such that $T_{i'j'}$ is a maximum.

EW3. If none of the constraints are violated by connecting i' to j' and disconnecting i' from the centre, then do so, set $T_{i'j'}:=\infty$ and goto EW2. Otherwise Set $T_{i'j'}:=\infty$ and goto EW2.

In the example of Fig 5.a, the Esau-Williams gives the solution shown in Fig 5.d, which is optimal. Chandy & Russell [1972] and Kershenbaum & Chou [1974] report that this method, in their tests, gave the best results of all those considered, exceeding the optimum cost by an average of less than 5%.

5.3.1.2.4. A Unified Algorithm

Kershenbaum & Chou [1974] showed that the three algorithms described above are all special cases of the same unified algorithm. This proceeds as follows.

U1. Connect all nodes i=1,2,..,n to the centre. Calculate w_i, i=1,2,..,n as described in I1. Let $T_{ij}=w_i-c_{ij}$ for each node pair.

U2. Find the pair (i',j') such that $T_{i'j'}$ is a maximum. If $T_{i'j'}<0$ then stop.

U3. If none of the constraints are violated by the addition of link (i',j') then goto U4. Otherwise set $T_{i',j'}:=\infty$ and goto U2.

U4. Connect nodes i' and j'. Disconnect i' from its previous connection. Update w_i and/or w_j as described in I2. Goto U2.

The initialisation and updating of the w_i are carried out according to the following set of rules.

I1. Initialisation.	Kruskal	: $w_i:=0$, $i:=1,2,..,n$.
	Prim	: $w_i:=-\infty$, $i:=1,2,..,n$.
	Esau-Williams	: $w_i:=c_{i0}$, $i:=1,2,..,n$.

I2. Updating (when link (i',j') is included).

Kruskal	: Nothing.
Prim	: $w_{j'}:=0$.
Esau-Williams	: $w_{i'}:=w_{j'}$.

The unified algorithm can be improved considerably by only considering the k nearest neighbours of each node as potential connections. This is entirely justified since it is rare that a node is connected to another a large distance from it. It can be shown (Kershenbaum & Chou [1974]) that the complexity of the algorithm is given by $An^2+Bkn+Ckn(\log_2 k)$, or $O(n^2)$ steps.

5.3.1.3. Extended Algorithms

The simple algorithms suggested above give reasonable solutions in a relatively short amount of time. Running times of $O(n^2)$ steps imply that quite large networks can be dealt with in this way, and with acceptable error in most cases.

For particularly large, and consequently expensive, networks, however, the saving in cost of only a few percent may make the apparently excessive cost of the exhaustive search method, running on a powerful machine, viable.

In cases where the choice to be made is not so clear, a third alternative exists: Hybrid and extended heuristic methods.

5.3.1.3.1. Extensions to MST Algorithms

Much use can be made of the observation (Chandy & Russell [1972]) that the set of nodes incident with the centre in the unconstrained (possibly infeasible) MST solution is a subset of the the set of similar nodes for the optimum constrained case. The following two-pass method provides an improvement to the simple case.

ET1. Construct an unconstrained MST on the n+1 nodes. If this is feasible then stop.

ET2. Select those nodes which are incident to the centre. Fix the appropriate links.

ET3. Construct a constrained MST on the n+1 nodes with the fixed links as a base.

In this way, some of the erroneous links included in the first pass, chosen before the fixed links, can be avoided in the second pass. The complexity of the algorithm, in terms of the O notation, has not increased.

A similar effect can be obtained by replacing ET3 by

ET3'. Implement the Esau-Williams algorithm on the n+1 nodes with the fixed links as a base.

A number of similar combinations are possible using, for example, the algorithms of Martin [1967] or Vogel (see Chandy & Russell [1972]). In tests, the majority of these methods give small average improvements, of less than one percent over the equivalent single pass technique.

5.3.1.3.2. Hybrid Algorithms

It has been demonstrated in a number of areas of problem solving (such as Kamel & Ismail [1983/4]) that where two, or more, methods exist for a particular application, a suitable combination of these methods will often produce a compound technique, capable of superior performance above its individual components. There are two distinct ways in which such a method can be constructed.

Firstly, each of the simple algorithms, in turn, can be implemented on the network in question. One, or more, of these algorithms will produce a solution of cost, no greater than each of the others. This becomes the required solution. Use of this compound technique on a number of networks will clearly produce cheaper networks, on average, than each of the individual components.

Alternatively, the compound algorithm can alternate between steps of the component algorithms. For example, the following simple algorithm uses Kruskal, Prim and Esau-Williams type tests in order to achieve a solution.

C1. Add the shortest feasible link that does not form a circuit.

C2. If there are no more links to add then goto C3. Otherwise add the shortest feasible link which connects a node to the existing centre based tree and does not create a circuit.

C3. Apply an Esau-Williams trade-off comparison to determine any links which can be reconnected via an alternative node. Reconnect any such nodes accordingly. If there are no more links to add then stop. Otherwise goto C1.

Clearly, there may be other versions of this compound algorithm using different building blocks and modified structures. The usefulness of such an approach appears to be that the strengths of each individual method are present within the main algorithm whereas each tends to compensate for the weaknesses of the others. Improvements of about one percent are average.

5.3.1.3.3. Second Order Greedy Algorithms

All of the simple heuristic methods for dealing with centralised communication networks, examined to date, have one thing in common. A decision is made as to which link to add, remove or replace on the sole basis of the greatest improvement attainable at that stage of the optimisation process. No thought is given to the effect of this action at a later stage and no facility exists for remedying an inappropriate action at a previous stage. If there are μ options, $\Theta_1, \Theta_2, \ldots, \Theta_\mu$, at each stage, producing improvements, $\delta_1, \delta_2, \ldots, \delta_\mu$, respectively, then any technique of the above form in which option Θ_x is chosen such that $\delta_x \geq \delta_y$ for all $y=1,2,\ldots,\mu$ is known as a <u>first order greedy algorithm</u> (FOGA). Such algorithms are considered from a different point of view in chapter 6.

Karnaugh [1976] suggests an improved use of these ideas, involving a control algorithm making repeated calls to a FOGA (in this case the Esau-Williams algorithm). This technique is called a <u>second order greedy algorithm</u> (SOGA). Such an algorithm, still attempts to maximise the improvement in cost but with a much broader view of the options available. Precise details of the algorithm are not given here but can be found in the paper.

Improvements of two to three percent have been encountered using these techniques although the algorithm does take somewhat longer to run. 400 nodes appears to be about the limit on large machines.

5.3.1.4. Comments on Centralised Network Design

This section briefly discusses ways in which the applicability of centralised networks can be broadened to cover situations not originally considered of importance. First, some possible generalisations are investigated.

5.3.1.4.1. Generalising the Techniques

The value of E, the maximum traffic level allowed on a single link, has been presented as constant over the whole network. In practice, this need not be the case. Each node pair (i, j) could have a threshold, E_{ij}, associated with it representing the maximum amount of traffic on that link alone. This would certainly be more appropriate in practice.

For each node pair, there could even be a number of thresholds, $E_{ij0}, E_{ij1}, E_{ij2}, \ldots, E_{ijB}$, in which case there would also need to be a number of costs, $c_{ij1}, c_{ij2}, \ldots, c_{ijB}$ where c_{ijb}, $1 \leq b \leq B$ (B the number of thresholds) is the cost of the link between i and j when the magnitude of the traffic carried by the link, (i, j), lies between E_{ijb-1} and E_{ijb}. It is reasonable to take $E_{ij0}=0$ for all $i, j=0, 1, 2, \ldots, n$.

The costs themselves, of course, are not, in practice, arbitrary entries in a matrix (or number of matrices if there are a number of thresholds); They are calculated from whatever formula or costing strategy is appropriate. The cost of a link does, however, increase in this discrete way with the addition of each extra circuit.

5.3.1.4.2. Applying the Techniques to a Wider Area

Initially, the notion of a centralised network was applied to computer interconnection structures (Schwartz [1977]). This is not unreasonable since the central node is most closely analogous to a central computer in some form of distributed user network. Therefore such systems are likely to comprise of 100% data transmission. Another name for these networks, under these circumstances is multipoint or multidrop networks.

There are ways, however, in which these centralised systems can be applied to a much more general situation. Within a tandem area, the dependent nodes (PABXs) all have to have links to the tandem. Usually, this is via a star local network. If, however, an all PCM network is insisted upon, and the amount of traffic flowing from nodes to tandem is small relative to the PCM link capacity, then a comparable form of traffic combination may be possible and savings in cost can be maximised by implementing the above algorithms. These algorithms would have to be modified, however, to take account of the costs incurred at the junction nodes. In fact, a better approach to this type of problem is contained in section 5.4.1.4.

Another, more realistic, area of possible use of these types of algorithms is within the PABX itself. Each traffic source/sink will be connected to the PABX by a single circuit. It is common practice, for reasons of increased efficiency and layout, for some of these circuits to share the same paths towards the PABX. No switching is required where one wire meets another and it involves about the same amount of work to route two single circuits along the same path as one. In addition, there may be a limit, in some cases, on the maximum number of wires along a single path (maybe for reasons as simple as lack of space). Thus the requirements of the PABX internal configuration relate exactly to the centralised networks described above and the corresponding algorithms can be used to find the most efficient arrangement.

5.3.2. Linear, Integer and Mixed Integer Programming

The wealth of techniques collectively known as linear programming (starting with Dantzig [1963] and most recently Karmarkar [1984]) has been applied to almost every conceivable area of problem solving. Network optimisation is no exception. Linear programming methods minimise or maximise expressions of the form $Z = \underline{c}^T\underline{x}$ subject to a system of equations, $A\underline{x} <, \leq, =, \geq, > \underline{b}$. In these expressions, $\underline{c}$ ($\underline{c}^T = (c_1, c_2, \ldots, c_n)$) is the cost vector, A ($= a_{ij}$) is an m x n matrix called the constraint matrix, $\underline{x}$ ($\underline{x}^T = (x_1, x_2, \ldots, x_n)$) is the vector of values to be determined and $\underline{b}$ ($\underline{b}^T = (b_1, b_2, \ldots, b_m)$) is the constraint value matrix. Z is simply the cost of the solution. Usually, by a suitable transformation of the variables or the addition of slack and surplus variables, the system of equations can be written in the form $A\underline{x} = \underline{b}$. A and $\underline{b}$ are known and $\underline{x}$ is to be determined. The inner product of $\underline{x}$ with each row of A, as well as the corresponding element of $\underline{b}$ is called a constraint. The function $Z = \underline{c}^T\underline{x}$, to be maximised or minimised, is called the objective function.

5.3.2.1. Methods of Solution

By far the most established technique for solving linear programming problems is the <u>simplex method</u> (see Fryer [1978] or Thie [1979]). The feasible solutions to any problem will lie within the boundaries of a polytope in n-D hyperspace. This polytope is defined as the enclosed space between the hyperplanes described by the constraints. The objective function will be represented by another hyperplane within the same dimension. As the values comprising the vector $\underline{x}$ vary, the hyperplane moves through space. The maximum or minimum feasible value of the objective function will occur when the hyperplane just touches the feasible polytope. This will occur at one of the vertices of the polytope. If the hyperplane is parallel to an edge, face, etc. of the polytope then an infinite number of points will touch simultaneously. The extreme point is still, however, a valid solution.

The simplex method operates by logically working round the vertices of the polytope, moving in the direction of the greatest improvement at each stage. When a vertex is found for which the objective function is more (or less - depending on whether the problem is to maximise or minimise) than for each of its neighbouring vertices, the process terminates. Even the simplest of methods is sophisticated enough to indicate single or multiple solutions, or (in the case of ill-defined problems) no solution.

There is, however, a considerable variation of implementation of the simplex method. In particular, the trade-off between the amount of work carried out at each step and the actual number of steps is seen to be extremely important in some cases (Mulvey [1978]). A (reasonably) recent development has been the algorithm of Karmarkar [1984] in which the feasible polytope is progressively transformed within hyperspace until a solution is found.

5.3.2.2. Extensions to the Simple Case

In the simple linear programming problem, the values $x_1, x_2, \ldots, x_n$ of the vector $\underline{x}$ can take any real value, i.e. $\underline{x} \in \mathbb{R}^n$.

Suppose, however, that $x_1, x_2, .., x_n$ are constrained to take integer values only; $\underline{x} \in \mathbb{N}^n$. The corresponding problem is known as an <u>integer programming problem</u>. The <u>integer</u> values of $x_1, x_2, .., x_n$ are required to maximise or minimise the value of Z.

If the variables $x_1, x_2, .., x_n$ can be partitioned into two sets X_1 and X_2 such that variables in X_1 must be integer, whereas the variables in X_2 may be any real, then the problem is a <u>mixed integer programming problem</u>.

A final variation upon the general theme can be made when the $\underline{x}$ vector is restricted to a form such that $x_i \in \{0,1\}$ for all $i=1,2,..,n$ (i.e. $\underline{x} \in \{0,1\}^n$. Such specifications give rise to <u>zero-one programming problems</u>.

A reasonable amount of work has been carried out with regard to these special problems and some success has been achieved (see Kaufmann & Henry-Labordere [1977]). In general, however, they are not as well solved as for the unconstrained problem and some instances are known to be NP-complete (Johnson & Papadimitriou [1985]).

5.3.2.3. Application to Network Optimisation

To describe the NOP as a programming problem again necessitates the assumption that certain costs remain constant. With this simplification, the method proceeds as follows.

The n nodes are labelled $i=1,2,..,n$ and each node is a potential tandem site. Let C_{1i} be the cost of a tandem at site i, let C_{2ij} be the cost of a link between node i and a tandem at j and let C_{3ij} be the cost of a link between a tandem at i and a tandem at j. Furthermore, let t_i be a variable which is 1 if there is a tandem at site i (0 otherwise), let x_{ij} be a variable which is 1 if there is a link between node i and a tandem at j (0 otherwise) and let y_{ij} be a variable which is 1 if there is a link between a tandem at site i and a tandem at site j. The problem can then be stated as follows.

$$\text{Minimise} \quad Z = \sum_{i=1}^{n} C_{1i} t_i + \sum_{i=1}^{n} \sum_{j=1}^{n} C_{2ij} x_{ij} + \sum_{i=1}^{n} \sum_{j=1}^{n} C_{3ij} y_{ij}$$

subject to the constraints

$$\sum_{i=1}^{n} t_i \geq 1$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i=1,2,..,n$$

$$\sum_{i=1}^{n} (y_{ij}-t_i/2) \geq 0, \quad j=1,2,..,n$$

$$x_{ij} \leq t_j, \quad y_{ij} \leq t_i, \quad y_{ij} \leq t_j, \quad i,j=1,2,...,n$$

$$t_i, x_{ij}, y_{ij} \in \{0,1\}, \quad y_{ij}=y_{ji}, \quad i,j=1,2,..,n$$

Z is the cost of the network. The first constraint ensures that there is at least one tandem in the solution network. The second set of constraints ensure that each node is connected to exactly one tandem. The third set of constraints ensure that the tandem network is connected. The fourth set of constraints ensure that nodes are only connected to tandems and that, within the tandem network, tandems are only connected to each other. The final set of constraints reveal the problem to be a zero-one programming problem.

The constraints ensuring full tandem connection are based on the assertion that a graph on M tandems is connected if each tandem is connected to at least M/2 other tandems. That this is true can be seen as follows.

Suppose M is odd. Consider tandem 1 and suppose, without loss of generality, that tandem 1 is connected to tandems 2,3,..,(M+3)/2. Suppose, contrary to the assertion, that there exists a tandem i, $(M+3)/2<i\leq M$ which has no path to tandem 1. Then tandem i cannot be connected to tandems 1,2,..,(M+3)/2, leaving a maximum of (M-5)/2

tandems available. This however, contradicts the requirement that each tandem be connected to at least M/2 others.

A similar argument applies for even M. Tandem 1 is connected to tandems 1,2,..,(M/2)+1, leaving a maximum of (M/2)-1 available.

This restriction, of course, excludes some valid tandem networks since there are a number of connected graphs which do not satisfy the third set of constraints. This is, however, the only way in which the necessary information can be expressed in linear form. It must be regarded as a limitation of the technique.

In principle, at least, the constrained optimum solution can now be found using one of the many linear programming software packages currently available or by self-implementation of any of the existing techniques.

A number of other linear, integer, mixed or zero-one programming formulations exist for simplifications of the general problem (Matsui [1978] and Hai-Hoc [1982a] & [1982b]).

5.3.3. The Restriction of Constant Costs

The formulations suggested in this section rely on the costs of the various components remaining the same under a variety of conditions of traffic loading etc.. In practice, of course, this is not the case. As discussed in the previous chapter, circuit and switch costs have a component which depends upon the amount of traffic they have to carry. The actual cost of a piece of equipment can thus only be determined if its size is known. In principle, it is possible to calculate these figures but, in practice, not at the start of such a process as required by these methods.

The cost of a tandem will depend upon how much traffic it has to switch. This in turn will be dictated by the closeness of the others. Tandems nearby will relieve some of the flow whereas a tandem on its own will need to carry more traffic. The positions of the transits are part of the required output from an NOP so information dependent upon then cannot be used as input to such a problem.

The cost of a link from a node to a transit will depend upon its length which, in turn, depends upon the position of its nearest transit. As before, such information is unavailable. Similarly, the tandem-tandem links will depend upon length and traffic capacity - information which is not known at the outset.

The areas of application of these methods is therefore somewhat limited. Better techniques are required to deal with more general cases in which costs vary almost continuously as the network topology changes.

5.4. Partial Solution Algorithms

Section 5.1.1.2 detailed the output that is required from any NOP solving procedure. This was given as a list of 10 items. Normally, the problem will be such that a solution cannot be obtained in this sequential form. Most of the output will be interrelated in the sense that the best solution for one section will depend upon the solution to another. A few obvious examples are as follows.

1. The optimum locations of the Ω tandems will depend upon the value of Ω.

2. The most suitable link type for each node-tandem connection will depend upon the size of the link and its length, the latter depending, in turn, upon the location of the tandems.

3. The optimum routeing strategy will depend upon the tandem network connection topology.

4. In turn, the most suitable link type for each tandem-tandem connection will depend upon the size of the link (which depends upon the routeing strategy) and its length (which depends upon the locations of the appropriate tandems).

Situations can arise, however, in which some aspects of the solution network are already decided upon. Much of the interplay between various sections of output may then not occur. Some examples of these situations are as follows.

1. The number of required tandems may already be specified by design. The choice of locations is then only a problem of selection of a fixed number of tandems.

2. The physical topology of the network would be established in a situation where an existing network is being redesigned. The routeing strategy is then a separate concern.

3. If all aspects of the design (tandem location, physical topology, routeing strategy, etc.) have been determined, the size, type and cost of the links etc. can be found by calculation.

This section considers solutions to some individual parts of the NOP. One area is discussed in detail (section 5.4.1) since this leads on to an extremely useful principle in a slightly different field of application. The remainder are considered briefly in section 5.4.2.

5.4.1. Tandem Location Methods

This section considers the problem of locating tandems within a network so as to make the most efficient use of the switching facilities available. There are two cases to consider; the number of tandems required may be known, or it may not.

If the number, Ω say, is known then the problem becomes one of finding the cheapest combination of Ω tandems from n nodes from all of the ${}^{n}C_{\Omega}$ available. This sort of problem will be dealt with, in passing, in the next chapter. In this section, the second case is assumed.

A valid solution to the NOP usually consists of a number of tandems selected from the original nodes. Tandems are not generally placed at any other site. In this section, however, the situation will be generalised to allow certain greenfield sites (i.e. sites not relating to original locations). In the unconstrained case, a tandem could, in principle, be placed anywhere. For the purposes of this section, a finite number of acceptable locations will be assumed.

Consider a situation with n nodes, $i=1,2,..,n$ and M potential tandem sites (greenfield or actual). The problem is to determine which of the sites should be open (in use) and which should be closed (not in use).

The next two sections discuss an approach to a restriction of this problem, in which constant costs and topological constraints are assumed. The following two sections seek to extend these arguments and apply the techniques to a somewhat different area.

5.4.1.1. The Add Algorithm

This section presents a version of the add algorithm first discussed by Bahl & Tang [1972]. It uses a remarkably simple approach to the problem which, by suitable adaptation, can be extended to a more general case.

To begin with, all potential tandems are taken to be closed save for a separate nominated one referred to as the centre. Every node is thus connected directly to the centre. The cost of opening a tandem j is given by F_j, $j=0,1,..,M$ where tandem 0 is the centre. F_j, $j=1,2,..,M$ includes the switching cost of the tandem together with the necessary link between itself and the centre. A maximum of X nodes may be parented onto any tandem, other than the centre. The cost of parenting a node i onto tandem j is c_{ij}, $i=1,2,..,n$, $j=0,1,..,M$.

The algorithm, in its most basic form is then as follows.

```
A1. Begin with all tandems closed.

A2. Open each tandem j in turn and

    A2.1. Find the set of Y nodes ($Y \leq X$) which produce the
          greatest improvement when connected to j.

    A2.2. Record this cost.

    A2.3. Close tandem.
```

A3. Select the tandem giving the greatest improvement on opening. If this improvement is non-positive then stop. Else open this tandem and reconnect nodes accordingly. Goto A2.

The improvement achieved in step A2.1 by reconnecting nodes i=1,2,..,Y (say) to tandem j is given by

$$\sum_{i=1}^{Y} (c_{ij'} - c_{ij}) - F_j$$

where j' represents the tandem to which each node i is connected before rerouteing.

As an example, consider the initial configuration shown in Fig 5.f. In this case X=4, giving effectively unlimited capacity to each tandem. Suppose $F_1=F_2=3$ and the connection costs are given by the following matrix.

$$C = \begin{array}{c|ccc|c} j = & 1 & 2 & 3 & \\ \hline & 10 & 4 & 7 & i=1 \\ & 7 & 6 & 3 & 2 \\ & 6 & 4 & 5 & 3 \\ & 3 & 9 & 5 & 4 \end{array}$$

Beginning with all of the nodes connected directly to the centre, the cost is given by Z=10+7+6+3=26. Opening tandem 1 produces the network shown in Fig 5.g, at a cost of Z=4+6+4+3+3=20 whereas opening tandem 2 gives the network in Fig 5.h, costing Z=7+3+5+3+3=21. Tandem 1 is thus chosen. The only possible opening at stage two is tandem 2. This gives the network of Fig 5.i, again at a cost of Z=4+4+3+3+3+3=20. This $\overset{is}{\wedge}$ identical to the cost of the network in Fig 5.g and suggests that either may be chosen. It would then be correct to make a choice on some form of reliability criterion.

There are a number of ways in which this algorithm can be generalised. The most obvious is to no longer regard the costs c_{ij} and F_j, i=1,2,..,n, j=0,1,..,M as constant. If the appropriate details of the network traffic are known then these costs can be determined as

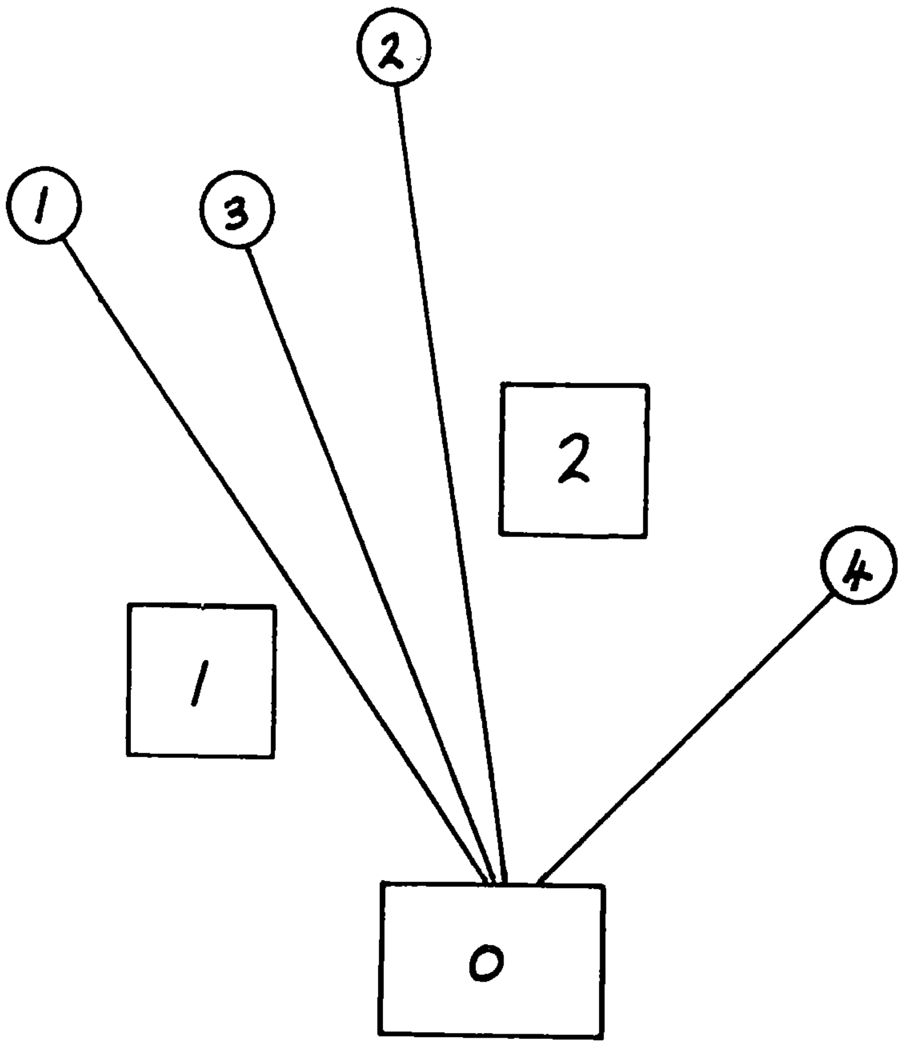


Figure 5.f. Add Algorithm: Initial Configuration

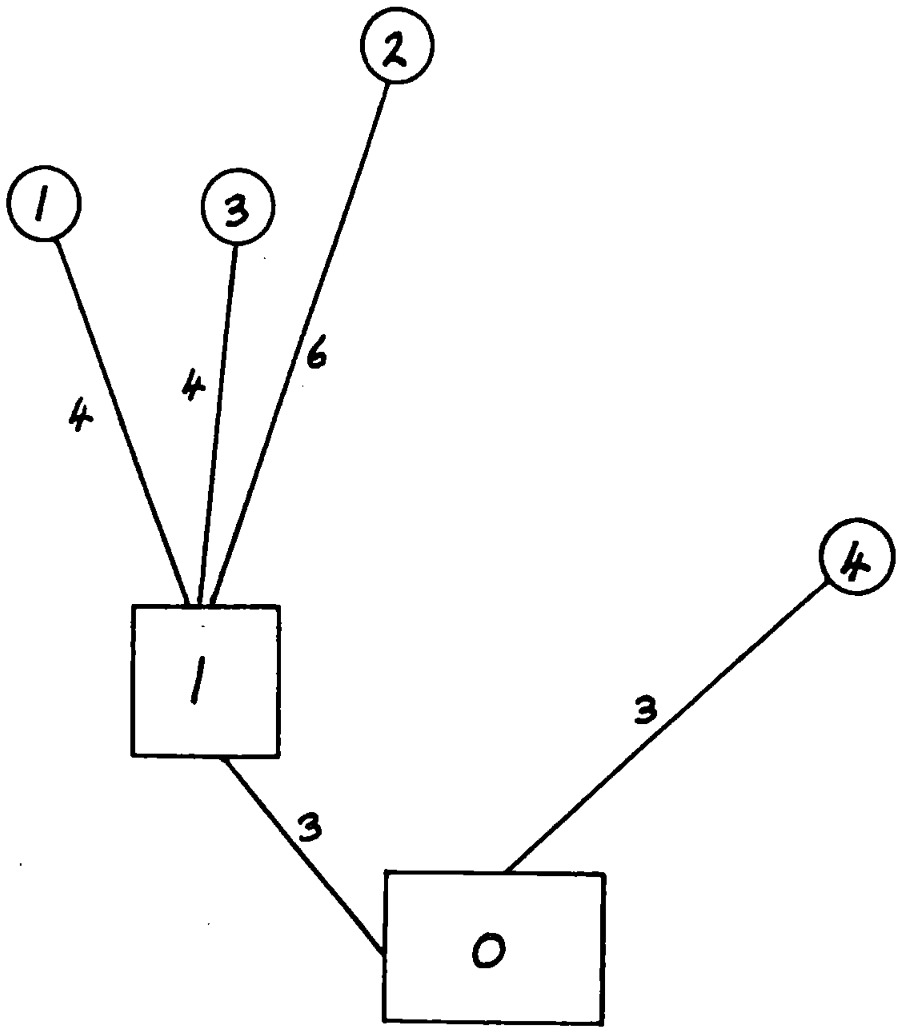


Figure 5.g. Add Algorithm: Tandem 1 Open

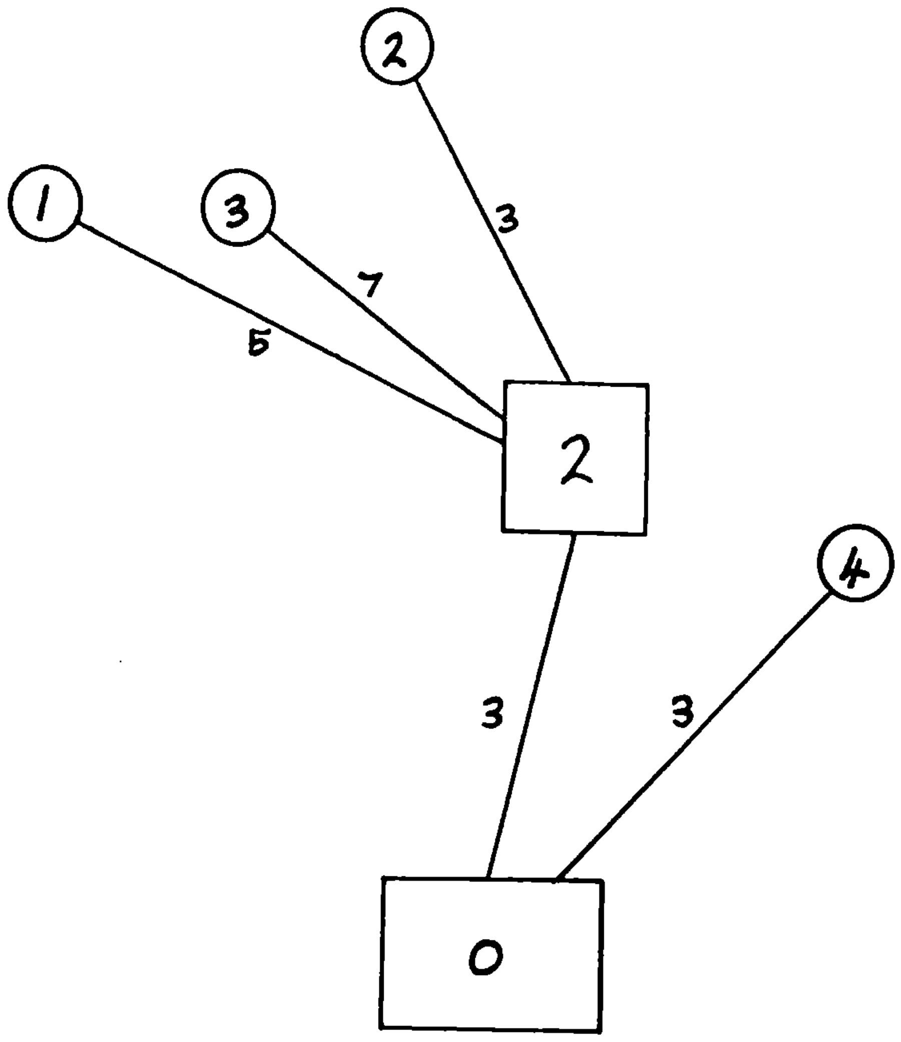


Figure 5.h. Add Algorithm: Tandem 2 Open

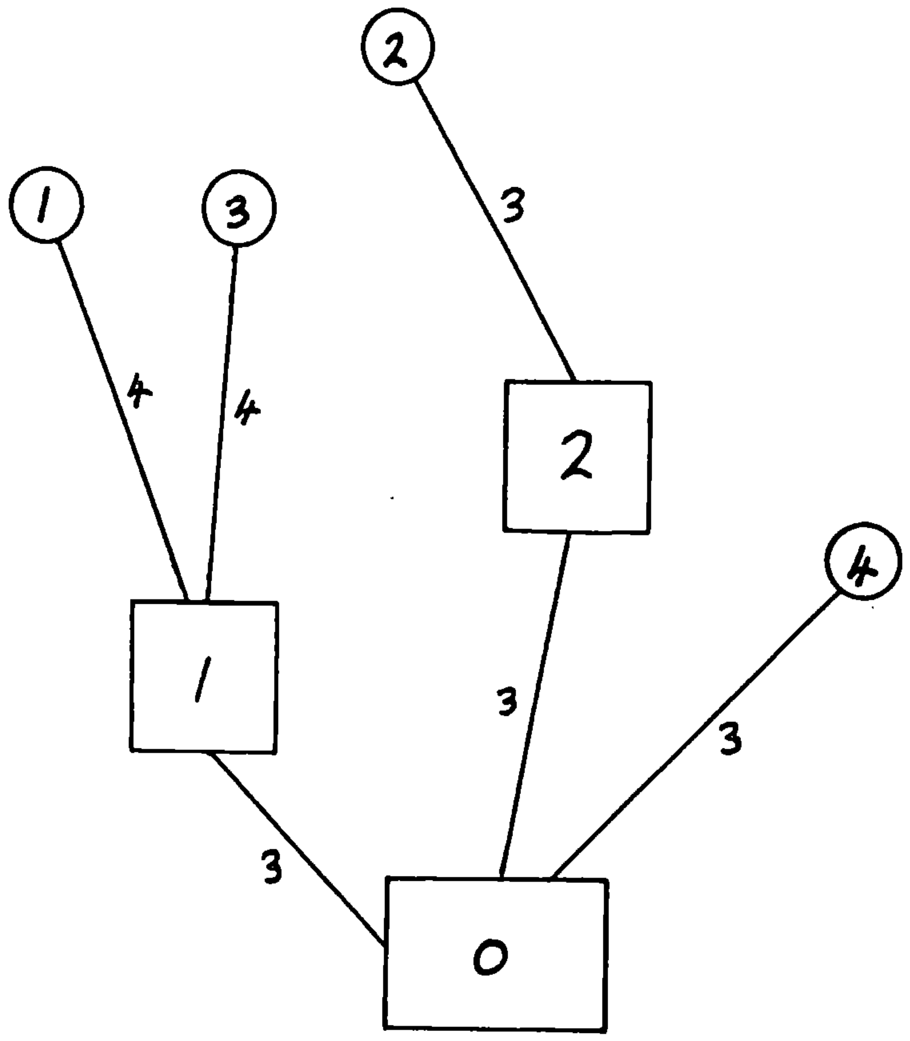


Figure 5.1. Add Algorithm: Tandems 1 and 2 Open

detailed in section 5.1.3. The tandem capacities can then be extended to a set of values X_j, $j=1,2,..,M$ where X_j represents the traffic capacity (in erlangs, etc.) of the tandem j. The process then operates as before. At each stage, every tandem is prospectively opened and those nodes which may be, legally and profitably, reconnected to it are so connected. The difference is merely that the new network needs to be formally recosted at each stage.

5.4.1.2. The Drop Algorithm

An obvious contrast to the add algorithm, in both its simple and extended forms, is the drop algorithm, again proposed in its most simple form by Bahl & Tang [1972]. As the name suggests, the algorithm starts with all tandems open and nodes connected, within the confines of the X constraints, to the nearest tandem. Each tandem is then systematically tried for closure. The tandem giving the greatest improvement on closure is duly closed and any dependent nodes reconnected, again within the confines of the X constraints, to the next nearest tandem. The process continues until no further improvement is possible. The add and drop algorithms are extremely well matched in terms of results.

5.4.1.3. An Extended Algorithm

The initial solutions employed as starting points by the add and drop algorithms (all tandems open or all tandems closed) are extreme. There is no particular justification for either and, in principle, a random starting solution would be just as acceptable (and, on average, would be closer to the optimum).

This observation leads to the following extended algorithm in which both the opening and closing of tandems is considered at each iteration of the process.

EAD1. Select $M/2$ tandems if M is even and $M/2+1$ or $M/2-1$ if M is odd from tandem sites $j=1,2,..,M$. Open these tandems. Close all others (apart from $j=0$). Connect each node to its

```
        nearest open tandem, subject to capacity constraints.

EAD2.   For each open tandem do the following

        EAD2.1. Try closing the tandem, as with the drop algorithm.

        EAD2.2. Record the cost of the resultant network.

        EAD2.3. Re-open the tandem.

EAD3.   For each closed tandem do the following

        EAD3.1. Try opening the tandem, as with the add algorithm.

        EAD3.2. Record the cost of the resultant network.

        EAD3.3. Re-close the tandem.

EAD4.   Select the tandem producing the largest improvement in
        cost. If this improvement is negative then stop.

EAD5.   Open or close the chosen tandem as appropriate. Reconnect
        nodes as appropriate. Goto EAD2.
```

In addition to starting from a better starting solution, this algorithm uses an improved modification system at each iteration. The strengths of the add and drop algorithms are combined. Whichever is to provide the greatest improvement will take precedence at each stage.

It should be noted that the add, drop and extended algorithms are all greedy in the first order (as discussed in section 5.3.1.3.3). Methods of improving upon this situation are discussed in the next chapter.

5.4.1.4. Local Network Structures

The add, drop and extended algorithms have, to date, been taken as acting upon a number of nodes and tandems in a hierarchical star configuration. Whilst such a structure is possible, it is not, in practice, common. A far more useful application exists, however, at a lower level of the network.

Consider a local tandem group. The tandem is connected into the rest of the tandem network in a particular (and for the purposes of this section, unimportant) topology. The nodes within the tandem group are to be connected to their parent tandem. The simplest method would be via a star. This, however, may not always be the cheapest.

A more general structure, particularly suited for large tandem groups, is to employ a number of <u>concentrators</u> as shown in Fig 5.j. A concentrator collects and distributes traffic within a given area and has a node (or tandem) as parent. This three level feature can be included in the network using any of the algorithms described in this section.

As an example, consider the add algorithm. Each of the nodes within a given tandem group is replaced by a concentrator parented directly onto the central tandem of that group. The add algorithm can then be used to systematically introduce nodes at those concentrator sites which produce the maximum improvement in cost.

This technique can be used, extremely easily, within the more powerful optimisation system detailed in the next chapter.

5.4.2. Other Individual Problems

The problem of tandem location is presented in some detail due to the importance of the techniques in their own right as well as the extension of these ideas to concentrator employment within a tandem group. Obviously, there are other problems to be solved and some of these may be susceptible to analysis in isolation. Considerable work has been done on a number of aspects of the complete problem. They are not particularly important in this context, however, and only two are discussed briefly here.

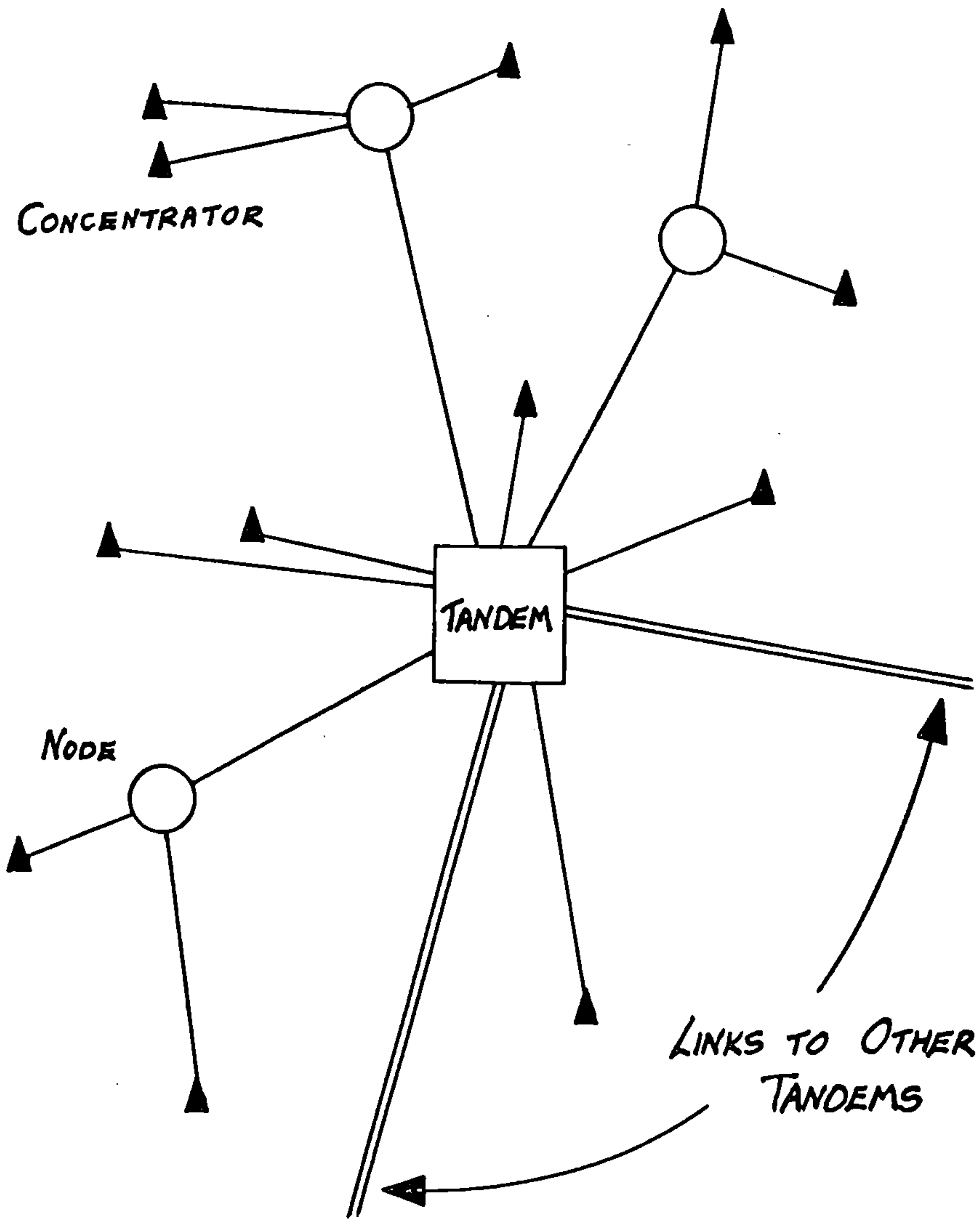


Figure 5.j. Local Tandem Network with Concentrators

5.4.2.1. Optimum Number of Tandems

The cost $C(N)$ of a network N can be regarded as a function $c(p_1,p_2,\dots,p_n)$ where the network N is described by the parameters $p_1,p_2,\dots,p_n$. The number of tandems M is one of these parameters. The optimum network N_M for a particular value of M has cost C_M which is a function $c(p_1,p_2,\dots,M,\dots,p_n)$ of the parameters describing N_M. The optimum value of M is then given by setting $\partial c/\partial M=0$.

If sufficient information is known about the likely characteristics of the optimum network in each case, it may be possible to estimate the function $c(p_1,p_2,\dots,M,\dots,p_n)$ by $c'(M)$ so that the cost of the optimum network with M tandems is taken as being entirely dependent upon M. If this is the case, then the optimum value of M can be approximated by setting $dc'/dM=0$ and solving for M.

A related problem, that of tandem area size, is considered by Hoshi [1985]. In this paper an attempt is made to determine the optimum group radius. Clearly the number of tandems and the average tandem area size are inversely interdependent. As is often the case, however, a number of simplifications and approximations are made in this paper which it is intended to avoid here.

5.4.2.2. Optimum Routeing Strategy

Given a particular tandem network, the problem of determining the best way in which to route traffic around it is one of the most difficult problems of network optimisation. The problem can, in principle, be broken down into subproblems for each pair of tandems. This requires that the best path between these two tandems be found. In practice, however, the choice of routeing for one tandem pair will affect the efficiency of various choices for another pair since a number of independent paths may, and in general will, use the same link in the tandem network. For exact solutions then, the routeing must be considered over the whole network at once.

The actual number of possible routeing strategies depends upon the topology of the tandem network. However, as discussed in section 5.1.2.2, there are unquestionably enough to prohibit exact search methods. Particularly stringent DORs may substantially simplify the problem but, in most cases, the number of possible solutions is still too large and approximate methods are required.

There are two distinct types of routeing problem encountered in network design and management problems.

1. Static routing problems. Such problems are encountered when it assumed that the strategy will not change from one time period to the next. This is generally the situation when networks are initially designed.

2. Dynamic routeing problems. These are problems in which the changing traffic characteristics between busy and slack hours are taken into account. Such problems are typical of situations in which a network already exists and only the best way of making use of it is unknown.

Within each category, there are, in principle, two further cases to consider. The primary route only between two tandems may be required, or a list of preferential routes may be necessary. Clearly, if a link within the primary path between two tandems fails or becomes overloaded then the secondary path must be known (and so on). In practice, however, this problem is usually dealt with at a later stage from the network design.

The purpose of this work is to design networks from the raw data (network design), not to optimise the operational efficiency of an existing one (network management). With this in mind, most references to traffic routeing will be intended to imply static routeing in which only the primary path is considered between any two tandems.

Various approaches to the different problems of traffic routeing have been suggested by, amongst others, Wallstrom [1969], Frank & Chou [1971], Yu et al. [1984] and Mars & Narendra [1987]. The problem is considered further in section 5.5 and, in a new light, in the next chapter.

5.5. Practical Methods of Optimisation

The methods discussed, to date, in this chapter belong, in the main, to the theoretical side of network optimisation. Assumptions and approximations are made with respect to the problem and its parameters which may only be proved valid under certain circumstances. Alternatively, some modification or extension may be required before such techniques can be implemented in a practical environment.

It is the purpose of the next chapter to introduce new techniques for practical network optimisation. Some of these will be extensions of existing methods but most will be developed from nothing. In this final section, however, some of the concepts involved in current network optimisation problems are introduced briefly. Firstly, the ideas and restrictions behind existing methods are discussed followed by a description of one of the more widely used methods.

5.5.1. Operational Techniques and Restrictions

The optimum solution represents the best network structure and traffic flow that can be achieved with no practical restrictions upon the final topology etc. but often such flexibility is not in evidence. A number of practical restrictions may be imposed upon the final solution for various reasons. Some are as follows.

1. Some tandem sites may be fixed by choice. Other nodes can be prohibited from housing tandems. This may be due to the equipment currently available or unavailable within each region or to an exterior knowledge of the likely change in traffic characteristics in the future.

2. For similar reasons, certain links may be essential or prohibited. Alternatively, links of a capacity exceeding that required may be used if traffic is expected to increase.

3. Routeing strategies between tandems can be dictated by practical restrictions also. The shortest distance (a set of

straight lines) between two points may pass through water or through a prohibited area. Often this will require that alternative paths are used.

As stated, these restrictions may be caused by constraints within the real world. They can, however, be introduced artificially into the problem in order to simplify the solution procedure. With reference to the three points mentioned above, simplifications can be achieved as follows.

1. The fixing of Ω' tandems from Ω decreases the number of possible tandem sets from ${}^{n}C_{\Omega}$ to ${}^{n-\Omega}C_{\Omega-\Omega'}$.

2. The fixing or prohibiting of L tandem-tandem links within the tandem network reduces the number of valid solutions by a factor of 2^{L}.

3. The saving in effort of restricting the permissible routeing strategies will depend, not only upon the level of restriction but also upon the particular tandem network topology under consideration.

These restrictions need not be entirely artificial. Often intuitive knowledge of the network environment can be used to supplant various areas of optimisation. It may be known from experience that traffic into and out of certain nodes is high. A reasonable (although still not fully justified) simplification is to fix tandems at these nodes along with the links connecting them. Direct traffic flow between these nodes is also evident.

When all simplifications have been made, however, optimisation, of some sort still needs to be achieved. The next section outlines one possible method of approaching this within a practical environment.

5.5.2. The Double Drop Method

There are a number of intuitive ways in which the optimisation of practical telecommunication networks may be achieved. This section presents one such approach. The algorithm is based, very loosely, upon the drop algorithm of Bahl & Tang [1972] (see section 5.4.1.2). It employs a form of <u>double drop</u> technique in which both tandems and links are sequentially removed, in a two-level nested-loop arrangement, until no further improvement can be found. The structure of the algorithm is as follows.

```
I:1 ;
make every node i a tandem ;
M :: n ;
REPEAT BEGIN
        maximum :: 0 ;
        connect each tandem to every other tandem ;
        route all intertandem traffic directly ;
        connect any non-tandem nodes to their nearest tandem ;
        calculate cost ;
        FOR j :: 1 TO M DO
             BEGIN
             remove tandem j {temporarily as a test} ;
             connect each tandem to every other tandem ;
             route all intertandem traffic directly ;
             connect any non-tandem nodes to their nearest tandem ;
             REPEAT BEGIN
                     max_saving :: 0 ;
                     FOR l :: 1 to (M-1)*(M-2)/2 DO
                          IF link l exists THEN
                             BEGIN
                             remove link l {temporarily as a test} ;
                             reroute traffic via cheapest
                             alternative ;
                             calculate saving ;
                             IF saving>max_saving THEN
```

```
                                  BEGIN
                                  max_saving :: saving ;
                                  l' :: l ;
                                  END ;
                              replace link l ;
                              END ;
                         IF max_saving>0 THEN
                            BEGIN
                            remove link l' {temporarily} ;
                            reroute traffic via cheapest alternative ;
                            END ;
                         END ;
                         UNTIL max_saving=0 ;
                calculate new_cost ;
                improvement_in_cost :: cost-new_cost ;
                IF improvement_in_cost>maximum THEN
                   BEGIN
                   maximum :: improvement_in_cost ;
                   k :: j ;
                   END ;
                replace tandem ;
                END ;
          IF maximum>0 THEN
             BEGIN
             remove tandem k {permanently} ;
             M :: M-1 ;
             END ;
          END ;
          UNTIL maximum=0 ;
REPEAT BEGIN
          max_saving :: 0 ;
          FOR l :: 1 to M*(M-1)/2 DO
              IF link l exists THEN
                 BEGIN
                 remove link l {temporarily as a test} ;
                 reroute traffic via cheapest alternative ;
```

```
                    calculate saving ;
                    IF saving>max_saving THEN
                       BEGIN
                       max_saving := saving ;
                       l' := l ;
                       END ;
                    replace link l ;
                    END ;
             IF max_saving>0 THEN
                BEGIN
                remove link l' {permanently} ;
                reroute traffic via cheapest alternative ;
                END ;
             END ;
             UNTIL max_saving=0 ;
      O=O ;
```

To begin with, a tandem is placed at every node. Each tandem is connected to every other tandem and all traffic flows along the direct links between them.

Each tandem, in turn, is removed from the tandem network (i.e. it is relegated to an ordinary node and connected to another tandem). The cost of the new network is calculated and, if cheaper than the original, the details are recorded.

Within each tandem removal phase, links are removed, one at a time until there is no further improvement. As each link is removed, the traffic on that link is rerouted via the cheapest alternative tandem. At the end of this process the cost of the new network is determined and, if cheaper, the details are recorded. This process is repeated for each tandem.

The tandem whose removal gives the greatest improvement is then removed permanently and the process is repeated until no further improvement is evident. When all unnecessary tandems have been removed, the link removal sequence is repeated for a final time, the removals now being made permanently.

This method will allow networks of a moderate size (100-200 nodes) to be dealt with on large machines. It represents the closest attempt yet at a practical method of optimising actual telecommunication networks. It still possesses the problems associated with greedy algorithms, however, since a tandem can easily be removed at an early stage which proves to be of value in the final solution. The next chapter presents a series of more sophisticated techniques.

CHAPTER 6

NEW TECHNIQUES FOR TELECOMMUNICATION NETWORK OPTIMISATION

The reasonable man adapts himself to the world: the unreasonable one persists in trying to adapt the world to himself. Therefore all progress depends on the unreasonable man.

George Bernard Shaw

Reason

Chapter 5 presents a number of different facets of the ordinary network optimisation problem. These are, in the main, concerned with certain simplifications and restrictions to the general case. Alternatively, methods were suggested which either take large amounts of time to run or use a drastically restricted optimisation approach.

This chapter presents a number of new techniques for various aspects of the NOP. Each of the sections which follow discusses one stage of, what will prove to be, a composite algorithm for the complete solution of the NOP. These sections relate roughly to the principal difficulties encountered in attempting such a solution.

1. In a practical situation, the number of nodes (n) in the initial network may be extremely large. The number of possible solutions to the NOP will increase rapidly as n increases. It will be noted that any technique which attempts to act directly on the full set of initial nodes will be eventually doomed to failure.

2. If the number of tandems in question is anything other than small, then the number of possible tandem sets, chosen from the initial set of nodes, will be of considerable proportions, even for lesser values of n. The problem of selecting the best one is extremely difficult indeed.

3. The number of valid tandem networks which can be constructed on Ω tandems is bounded below by $2^{\Omega(\Omega-1)/2-1}$. Choosing the best presents considerable difficulties.

4. As mentioned in the previous chapter, the problem of determining the optimum routeing strategy is possibly the most involved problem of all. The number of permissible flows is enormous (dependent upon the topology).

Various heuristic methods are introduced in this chapter for overcoming each of these problems in turn.

A method is presented in section 6.1 for dealing with the problem of very large networks involving the replacement of the initial network by a representative one, allowing simplified optimisation to take place.

Sections 6.2 and 6.3 present methods for selecting the required set of tandems from the original nodes, firstly by restricting the form of the tandem network and then by applying a series of local transformations (perturbations) to this initial solution.

The problems of determining the tandem network connection topology and the traffic routeing strategy are combined and discussed in section 6.4, using a link removal technique with lookahead capabilities. The relationship between perturbation and lookahead is then considered.

Finally, section 6.5 outlines a complete method for the solution of the NOP based upon the component techniques presented in the previous sections.

6.1. Reduction Techniques

Practical networks may need to interconnect many hundreds of nodes. To consider each node individually within any optimisation technique would produce an algorithm of ridiculously long running time (as can be seen by comparison with the TSP, solutions for which are not easily obtained for more than a few dozen cities - see section 5.2.3). Some form of effective heuristic is clearly required.

6.1.1. A New View of the Problem

The input to an NOP consists, amongst other things, of the details of n nodes. The time taken to solve the NOP, using the best known exhaustive search algorithm can be considered as a function of this number of nodes, $T=f(n)$. The problem, in these terms, is that T becomes too large - too quickly as n increases.

The purpose of any conceivable heuristic technique will be to decrease the value of T, in any particular case, to a new time U where $U<<T$. There are, in theoretical terms, exactly two mutually exclusive ways of achieving this.

1. Transform the function f to a less severely increasing function g (i.e. $[df/dn]_{n=n'}>[dg/dn]_{n=n'}$ for all n'). Then $f(n)>g(n)$ and g can be chosen such that $T=f(n)>>g(n)=U$.

2. Replace the parameter n by a smaller parameter q. Then $f(n)>f(q)$ and, for larger values of n, $T=f(n)>>f(q)=U$.

The first option corresponds to the case in which a simplified optimisation procedure operates upon the original set of nodes. All techniques discussed to date have been of this form. Some set of assumptions and/or approximations is made in order to deal with large numbers of nodes. Such methods can be referred to as <u>simplification methods</u>.

The second form of approach uses the exact optimisation algorithm but on a reduced number of nodes. The solution obtained will be exact with respect to this smaller set but, for this solution to be of any use, the reduced set must somehow represent the original. Techniques of this type will be referred to as <u>reduction methods</u>. Section 6.1.2 presents a method for obtaining this reduced network.

There are, of course, other options involving combinations of these two extreme forms. Typically, a large network could be replaced by a smaller one which is still too large to act upon with an exhaustive search algorithm. An accurate heuristic method may now be used, however, which could not have been used upon the original network. The exact trade-off value between reduction and optimisation will be dependent upon the problem in hand and the reduction and optimisation techniques it is intended to use. The composite method suggested at the end of this chapter is actually of this hybrid form.

6.1.2. Reducing Networks

The concept of replacing an unmanageably large network with an acceptable smaller one could easily relate to a number of problem areas in which immediate solution is prohibited by the number of parameters (see chapter 8). A precise arrangement is now suggested for telecommunication networks in particular.

Firstly the objectives of such a method are outlined. The details of the actual method are then given, followed by comments upon how to overcome certain difficulties in the reduction process.

6.1.2.1. Objectives

If the NOP were to be solved in a series of steps, as opposed to being solved as a whole (exactly or heuristically, in a manner suggested implicitly by the algorithm of section 5.2.1), then the most natural starting point would be the determination of the Ω nodes, from the original n, most suitable to act as tandems within the solution network (since it is unreasonable to discuss connection topologies and routeing strategies before their positions are known). As noted in section 5.1.2.2 the number of such selections ($\Sigma_{M=1..n}[n!/(M!(n-M)!)]$) prohibits the explicit testing of each. Some form of simplification is required.

A method is presented which replaces the original n node starting network with a representative one of q nodes. These q nodes must be chosen in such a way as to contain, in simplified form, sufficient information regarding the distribution of nodes and traffic of the original. If this is the case then exact methods or accurate heuristics can then be used in order to choose the required set of Ω tandems. Sections 6.2 and 6.3 discuss this tandem selection process.

6.1.2.2. Step by Step Reduction

The reduction process operates in a step by step fashion. The original n node network is replaced by an n-1 node one, which ,in turn, is replaced by an n-2 node one, and so on until the replacement q node network is reached. At each stage, a p node network is replaced by a p-1 node network (p=n,n-1,..,q+1).

Prior to the commencement of the reduction process, <u>weights</u> w_i, i=1,2,..,n must be assigned to each of the nodes These weights are a measure of the importance of a node and should be calculated in such a way as to reflect the traffic being generated and received by it (see section 6.1.2.3.). Each step of the reduction process is then implemented as follows.

At each stage, the locations of p nodes are known as well their weights. The object is to produce a p-1 node network representing the original.

```
[SINGLE REDUCTION STEP] ;
I:={p, $x_1, y_1, w_1, x_2, y_2, w_2, \ldots, x_p, y_p, w_p$} ;
minimum := $\infty$ ;
FOR i := 1 TO p-1 DO
    FOR j := i+1 TO p DO
        IF $\surd[(x_i-x_j)^2+(y_i-y_j)^2]$<minimum THEN
           BEGIN
           minimum := $\surd[(x_i-x_j)^2+(y_i-y_j)^2]$ ;
           i' := i ;
           j' := j ;
           END ;
p' := p-1 ;
$x_{i'}$ := $(w_{i'}.x_{i'}+w_{j'}.x_{j'})/(w_{i'}+w_{j'})$ ;
$y_{i'}$ := $(w_{i'}.y_{i'}+w_{j'}.y_{j'})/(w_{i'}+w_{j'})$ ;
$w_{i'}$ := $w_{i'}+w_{j'}$ ;
FOR i := j' TO p' DO
    BEGIN
    $x_i$ := $x_{i+1}$ ;
    $y_i$ := $y_{i+1}$ ;
```

```
        $w_i$ := $w_{i+1}$ ;
        END ;
    O:={$p'$, $x_1$, $y_1$, $w_1$, $x_2$, $y_2$, $w_2$, ..., $x_{p'}$, $y_{p'}$, $w_{p'}$} ;
```

The essence of the single reduction step is that the closest two nodes are replaced by a single node which represents them, both in terms of location and traffic loading characteristics. For example, in Fig 6.a, the nodes i and j are replaced by a node r where $x_r=(w_ix_i+w_jx_j)/(w_i+w_j)$, $y_r=(w_iy_i+w_jy_j)/(w_i+w_j)$ and $w_r=w_i+w_j$. The p-1 node network adequately represents the original in node and traffic distribution.

With this step defined, the complete reduction process is as follows.

```
I:={$n$, $x_1$, $y_1$, $w_1$, $x_2$, $y_2$, $w_2$, ..., $x_n$, $y_n$, $w_n$} ;
p := n ;
WHILE p>q DO
    BEGIN
    implement SINGLE REDUCTION STEP ;
    p := p' ;
    END ;
O:={$q$, $x_1$, $y_1$, $w_1$, $x_2$, $y_2$, $w_2$, ..., $x_q$, $y_q$, $w_q$} ;
```

The entire n node network is now represented by the smaller q node network. There are, however, a number of practical difficulties to overcome, as discussed in the next three sections.

6.1.2.3. Determining the Node Weights

The weight of a node should be a measure of its importance. When two nodes are replaced within the single reduction step, the resultant node will be drawn towards the more heavily weighted of the two. Such a weight must be calculated in terms of the traffic handled by the node. This calculation, however, is not as obvious as it might, at first, seem. Merely summing the incoming and outgoing traffic has the effect of exaggerating a heavily weighted node.

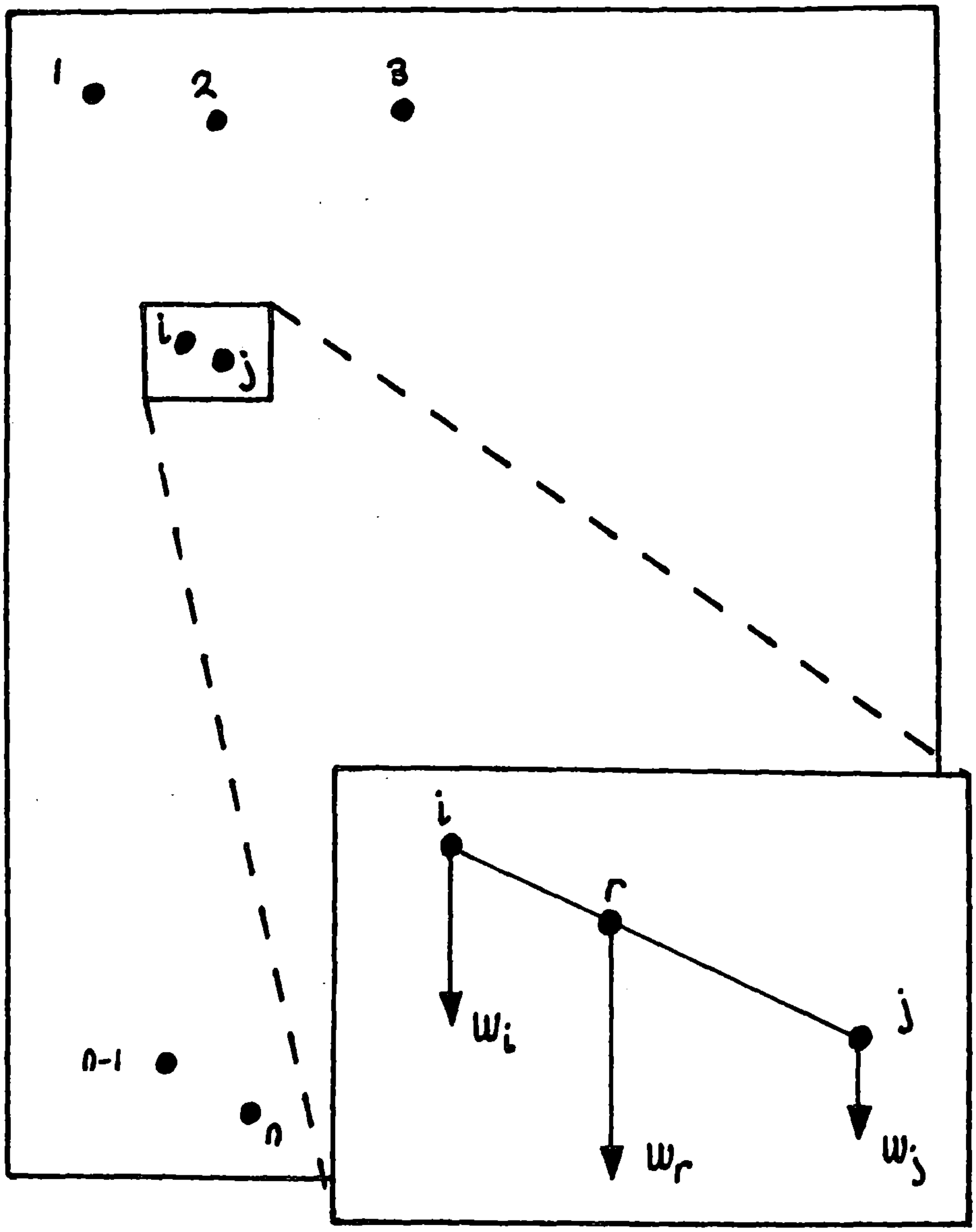


Figure 6.a. Single Reduction Step

A replacement node, r, will best represent its two components, i and j, if it costs the same to connect it to an imaginary tandem as would the original two nodes. This will be approximately achieved if the number of connecting circuits required by r is the sum of the circuits required by i and j. Since the weight of r is given by the sum of the weights of i and j, it follows that the best way to determine the weights at the beginning of the process is to let the weight of each node be proportional to the number of circuits required to carry the traffic into and out of the node.

Other weights are possible, of course, such as logarithmic or exponential functions of the sum of the traffic. Some function of the traffic should always be used.

6.1.2.4. Reducing the Traffic Matrices

At each stage, when a p node network is replaced by a p-1 node network, the corresponding traffic matrices must be reduced as well. The same operation will apply to all such matrices. For the purposes of demonstration then, suppose that only one such matrix, $A=(a_{ij})$, $i,j=1,2,..,p$, exists. A typical (but small) example is illustrated in Fig 6.b.

Suppose nodes i' and j' are replaced by a new i'. The corresponding rows and columns for i' and j' must be combined as well.

```
I={A,i',j',p} ;
$a_{i'i'} := a_{i'i'}+a_{i'j'}+a_{j'i'}+a_{j'j'}$ ;
FOR i := 1 TO p DO
    IF NOT ((i=i') OR (i=j')) THEN
       BEGIN
       $a_{ii'} := a_{ii'}+a_{ij'}$ ;
       $a_{i'i} := a_{i'i}+a_{i'j}$ ;
       END ;
p := p-1 ;
FOR i := j' TO p DO
    FOR j := 1 TO p DO
        BEGIN
```

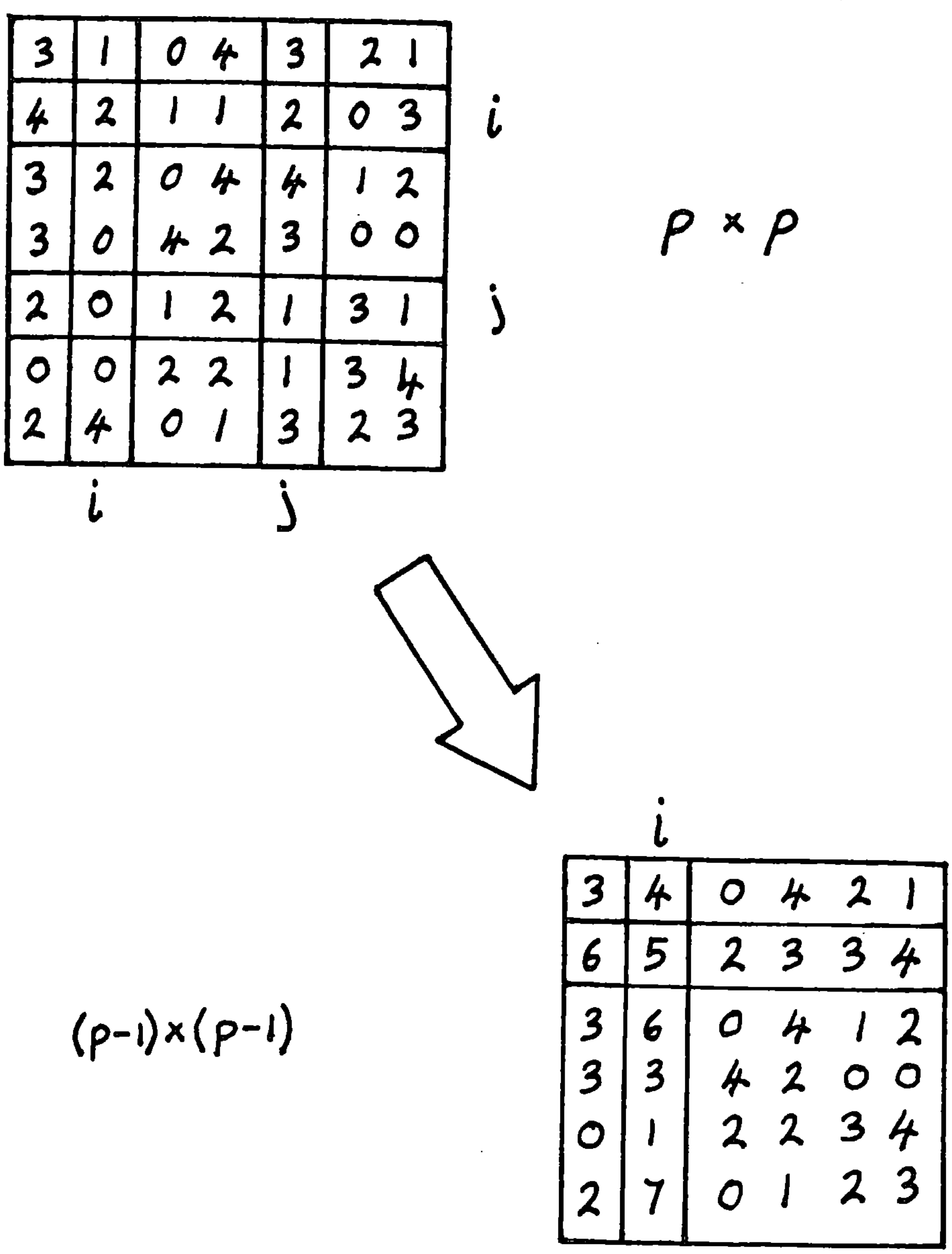


Figure 6.b. Reducing the Traffic Matrix

```
        $a_{ij}$ := $a_{i+1j}$ ;
        $a_{ji}$ := $a_{ji+1}$ ;
        END ;
O={A,p} ;
```

There is, however, an alternative. In practice, the traffic matrix is not actually required at each stage of the reduction process, only on its completion. This simplifies the task considerably. Instead of reducing the matrix step by step in parallel with the node reduction, the operation can be carried out in one stage at the end. Suppose an n node network (located at ($\underline{x}$,$\underline{y}$)) has been reduced down to a q node network (located at ($\underline{x}'$,$\underline{y}'$)) and a q x q traffic matrix, A', is required from the original n x n matrix, A.

```
I={A,n,q,$\underline{x}$,$\underline{y}$,$\underline{x}'$,$\underline{y}'$} ;
A' := $\underline{0}$ ;
FOR i := 1 TO n DO
    BEGIN
    minimum := $\infty$ ;
    FOR j := 1 TO q DO
        IF $\sqrt{[(x_i-x'_j)^2+(y_i-y'_j)^2]}$<minimum THEN
        BEGIN
        minimum := $\sqrt{[(x_i-x'_j)^2+(y_i-y'_j)^2]}$ ;
        closest_to[i] := j ;
        END ;
FOR i := 1 TO n DO
    FOR j := 1 TO n DO
        $a'_{closest\_to[i]closest\_to[j]}$ :=
                                   $a'_{closest\_to[i]closest\_to[j]}+a_{ij}$ ;
O={A'} ;
```

The closest of the q replacement nodes is determined for each of the original n nodes. The traffic is then grouped accordingly.

6.1.2.5. Greenfield Sites

The nature of the reduction process is such that the vast majority of the q replacement nodes, if not all, will be located at greenfield sites (see section 5.4.1). Choosing a set of nodes to act as tandems will therefore result in a number of tandems whose positions do not correspond to any of the originals. If it is required that only actual node sites are to house tandems then one of two options is available.

1. Move each of the q replacement nodes to its nearest actual site after reduction and before the exact optimisation method or accurate heuristic is used.

2. Implement the exact method or accurate heuristic using the greenfield sites as tandem locations. At the completion of the process, move each tandem to its nearest actual site.

Results suggest that neither approach is substantially better than the other.

A third possibility is available, in theory, by calculating the new coordinates $x_{i'}$ and $y_{i'}$, at each stage of the reduction process, as

$x_{i'} := \max(x_{i'}, x_{j'})$;
$y_{i'} := \max(y_{i'}, y_{j'})$;

in the single reduction step. This, however, is not entirely satisfactory since an effectively arbitrary choice is made between two nodes of similar weighting which could be quite some distance apart.

6.1.3. Analysis of Reduction Techniques

Each node in the q node replacement network is intended to represent, in both (traffic) size and location, one or more of the original nodes. Size is not a immediate problem since, by definition, each node is equivalent, in terms of traffic weighting, to the two nodes it replaced at the previous stage. Errors in this reduction technique will be restricted to cases in which replacement nodes are to be found in inappropriate places within the reduced network (although this can, ultimately, also cause a node to be of an incorrect size). This section analyses theoretically the likelihood of such errors, together with their expected magnitude and consequences. Chapter 7 contains a selection of practical results.

6.1.3.1. Occurrence of Incorrect Reduction

This section presents a discussion of ways in which possible errors can arise in the reduction process. There are two intuitive ways in which locational errors can occur. Firstly, a replacement node may not, in principle at least, be located at the correct centre of the group of nodes it represents and secondly, the node may actually be representing the wrong group. It will be seen that the first case is impossible but the second is not.

6.1.3.1.1. Position of Replacement Nodes

Suppose a node I (with coordinates (X,Y) and weight W) has replaced nodes $i=1,2,..,\mu$ (each with coordinates (x_i,y_i) and weight w_i) in the reduction process. Then clearly

$$W = \sum_{i=1}^{\mu} w_i$$

as required, but it may be that X and Y are incorrect.

The ideal location (X',Y') of a node representing nodes $i=1,2,..,\mu$ will be at the circuit weighted centre of the group. Taking moments about the centre gives

$$\sum_{i=1}^{\mu} w_i(x_i-X') = 0 \qquad \text{and} \qquad \sum_{i=1}^{\mu} w_i(y_i-Y') = 0$$

Rearranging, gives

$$X' = \frac{\sum_{i=1}^{\mu} w_i x_i}{\sum_{i=1}^{\mu} w_i} \qquad \text{and} \qquad Y' = \frac{\sum_{i=1}^{\mu} w_i y_i}{\sum_{i=1}^{\mu} w_i}$$

It can be shown that $X=X'$ and $Y=Y'$. The proof for X is as follows, the case for Y being entirely similar.

The proof is by induction on μ. For $\mu=1$, $X' = w_1 x_1/w_1 = x_1 = X$ as required. Now suppose the proposition is true for all $\mu \le k-1$. Then for $\mu=k$, by the reduction process, $X = (w_\alpha x_\alpha + w_\beta x_\beta)/(w_\alpha + w_\beta)$ where node α has replaced nodes $1,2,..,p$ and node β has replaced nodes $p+1,p+2,..,k$. But since $1 \le p, k-p \le k-1$, the inductive hypothesis gives

$$X = \frac{\dfrac{\sum_{i=1}^{p} w_i \sum_{i=1}^{p} w_i x_i}{\sum_{i=1}^{p} w_i} + \dfrac{\sum_{i=p+1}^{k} w_i \sum_{i=p+1}^{k} w_i x_i}{\sum_{i=p+1}^{k} w_i}}{\sum_{i=1}^{p} w_i + \sum_{i=1}^{k} w_i}$$

$$= \frac{\sum_{i=1}^{k} w_i x_i}{\sum_{i=1}^{k} w_i} = X'$$

Clearly then, a replacement node for a number of initial nodes is placed, by the reduction process at the correctly weighted centre of the group and is given the correct weighting. This eliminates one possible source of error.

6.1.3.1.2. Node Groupings

Consider the reduction process from a different point of view. Each node i can be thought of as starting as the unique member of a set I_{i1}. As the process progresses, the node i (usually) becomes combined with another node j. In truth, the nodes i and j are replaced but they can be considered as being combined to form a new set I_{i2} ($\equiv I_{jx}$ for some x). The node i can thus be thought of as belonging successively to the sets $I_{i1}, I_{i2}, \ldots, I_{iF}$ until the reduction is completed. For some nodes i and j, $I_{iF} \equiv I_{jF}$ since each group may contain more than one original node. If there are q nodes in the replacement network then there are q unique sets from the n so defined and each node i belongs exclusively to exactly one of these sets. Relabel these sets $G_1, G_2, \ldots, G_q$.

Each of the original nodes will be closest to one particular replacement node. Define the sets $D_1, D_2, \ldots, D_q$ to be such that $i \in D_j$ if and only if j is the closest replacement node to i. If the reduction process functions perfectly, in any given case, then the sets $D_1, D_2, \ldots, D_q$ will partition the set of n nodes in exactly the same way as the sets $G_1, G_2, \ldots, G_q$. However, this is not necessarily the case. These sets can be different if one or more nodes have been drawn away from their groups.

Fig 6.c shows a typical case. The correct location of (some of) the replacement nodes are indicated (although they are obviously not known at the start of the reduction process). These nodes define a number of area groups, based upon shortest distance. These groupings relate to the sets $D_1, D_2, \ldots, D_q$. There is exactly one way in which an error can occur during the reduction process. Consider two nodes i and j such that $i \in D_\alpha$ and $j \in D_\beta$ ($\alpha <> \beta$). If i and j (or the nodes which may have replaced them) are combined (replaced) at any stage then an error will result. Either i will pull j into the set G_α or j will pull i

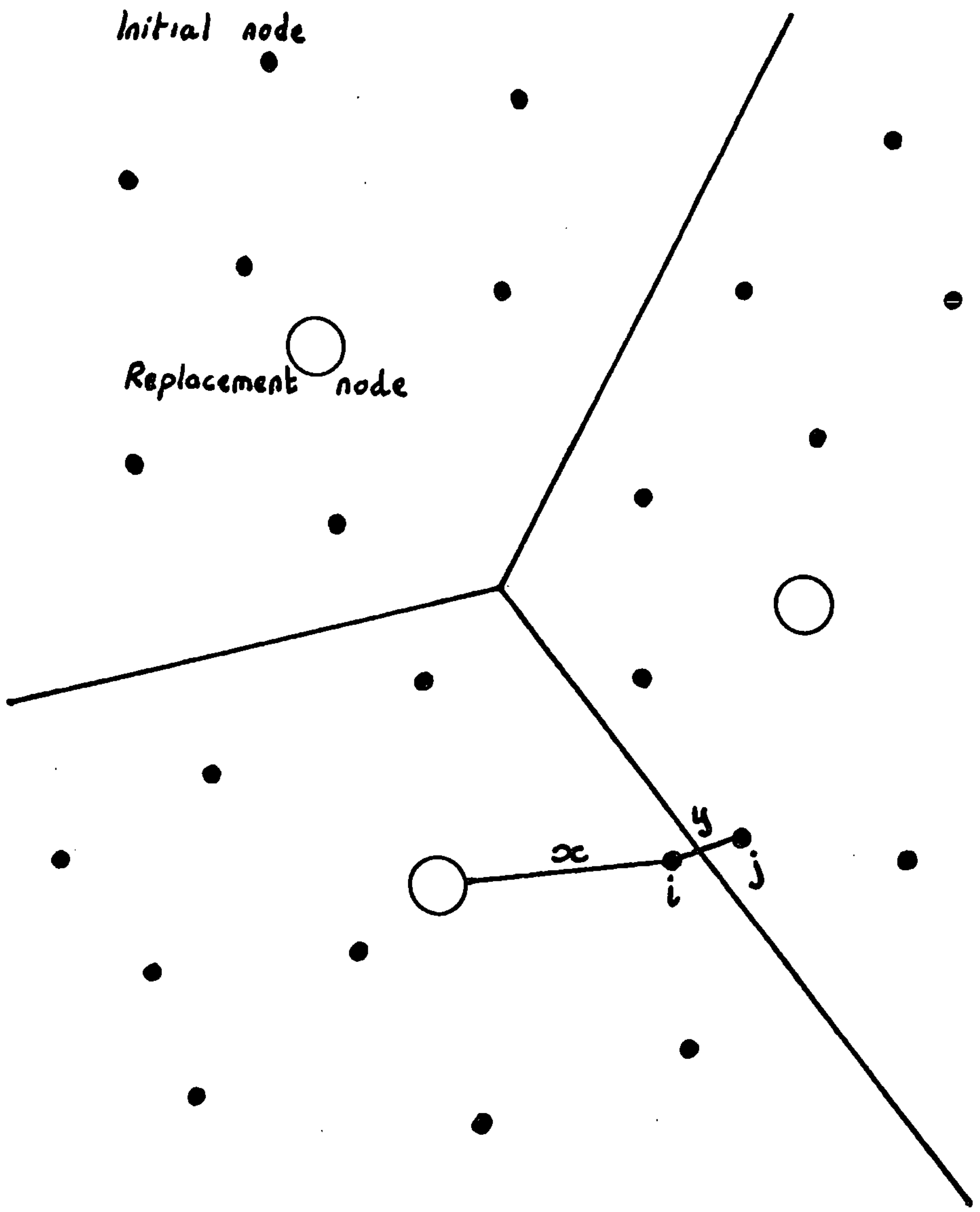


Figure 6.c. Incorrect Node Grouping during Reduction

into the set G_β. i and j will remain together from this moment on so even subsequent movements cannot correct the mistake - at least one of them will be in error.

The immediate problem is to determine the likelihood of this happening and, with this in mind, it is advantageous to express the problem in a slightly different form. Label each of the reduction stages n'=n,n-1,n-2,..,q+2,q+1 (stage n' reduces n' nodes to n'-1). At each stage n' the closest two nodes (i and j) are combined. The probability of an error is the probability that these nodes are in different groups. Denote this probability by P(n'). Then if r is the radius of the group containing node i, this probability is given by

$$P(n') = \int_0^r f(x) p_{n'}(x)\, dx$$

where f(x) is the <u>probability density function</u> (PDF) of the distance of node i from the centre of the group and $p_{n'}(x)$ is the probability, at stage n', that the nearest neighbour to a node, a distance x from the centre, is outside of the group. In turn, $p_{n'}(x)$ can be seen to be of the form

$$p_{n'}(x) = \int_0^\infty g_{n'}(y) b(x,y)\, dy$$

$g_{n'}(y)$ is the PDF of the distance between the two nearest nodes at stage n' and b(x,y) is the probability that a node at a distance y from a node at a distance x from the centre of the group, is not in the same group.

The probability of at least one error during the entire reduction process is given by

$$P = 1 - \prod_{n'=q+1}^{n} (1-P(n'))$$

In addition, the probability P(n') can be interpreted as the expected number (fraction) of a node to be in error at each stage. Summing over all stages gives the expected number of nodes in error from start to finish.

$$E = \sum_{n'=q+1}^{n} P(n')$$

In full these values are given by

$$P = 1 - \prod_{n'=q+1}^{n} \left(1 - \int_0^r f(x) \int_0^\infty g_{n'}(y) b(x,y)\, dy\, dx \right)$$

$$E = \sum_{n'=q+1}^{n} \left(\int_0^r f(x) \int_0^\infty g_{n'}(y) b(x,y)\, dy\, dx \right)$$

If it happens that one or more nodes are in error, then there are a number of minor effects. Firstly, the replacement nodes will be of the wrong size. One will be too large and another will be too small by an amount equal to w_j, the size of the node, j, which is incorrectly grouped. In addition, the relevant replacement node will be incorrectly located by a distance

$$\frac{\sum_{i=1}^{\mu} w_i x_i}{\sum_{i=1}^{\mu} w_i} - \frac{\sum_{i=1\ i<>j}^{\mu} w_i x_i}{\sum_{i=1\ i<>j}^{\mu} w_i}$$

where μ is the number of nodes in the group (including j). These two types of error may have an effect upon the subsequent progress of the reduction method and the eventual choice of tandems from these replacement nodes. The next section suggests a way of estimating this error in a practical case - the precise evaluation being unreasonable.

6.1.3.2. Estimating the Error in a Given Case

A useful resource, in any given case of network reduction, would be the facility to estimate the probability of any nodes being in error and the expected number of such nodes (P and E). This section presents some techniques for achieving this. The three component functions are discussed individually, followed by their combination in the calculation of P and E.

Some initial calculations are required, however. In particular, the area covered by the network and the radius of the replacement node groups. In certain cases, these may be known. If this is not the case, some work must be done to evaluate them. Since the nodes lie in the plane, their convex hull can be found (Graham [1972] and Shamos [1978]) and the area enclosed by it, A, evaluated. This can be achieved in O(nlogn) steps.

A different technique, also of complexity O(nlogn), is suggested here. This is only approximate but has the advantage of requiring data which is also needed at a later stage of the evaluation of P and E. Consequently, no extra work is required.

The work that is to follow is based, to some small extent, on a branch of statistics called <u>nearest neighbour analysis</u>. A comprehensive treatment of this field is not given here as the vast majority of the theory is unnecessary (Kendall & Moran [1963] and Shamos [1978]) and its applications, to data, have been somewhat obscure (Cottam & Curtis [1949] and Morisita [1954]).

Define n_{ij}, $i=1,2,..,n$, $j=0,1,..,n-1$ to be the node which is the j^{th} nearest neighbour to node i (where $n_{i0}=i$ by convention). Let ∂_{ij} be the distance of n_{ij} from i. Define the average distance of the j^{th} nearest neighbour in the network to be

$$\Delta_j = 1/n \sum_{i=1}^{n} \partial_{ij} \qquad j=0,1,..,n-1$$

This forms a discrete distribution over the values $j=0,1,..,n-1$ which can be used to describe the characteristics of the node arrangement (widely distributed, heavily clustered, etc.). These techniques are employed in sections 6.1.3.2.1 and 6.1.3.2.2 in the approximation of the functions f and $g_{n'}$. Similar problems of cluster detection are discussed by Zahn [1971].

The ∂_{ij} values can be used in a number of ways to approximate the area of the network, A, and the average group radius, r. One way (probably the simplest) is as follows.

Select a node i' such that $\partial_{i'n-1}=\max_i(\partial_{in-1})$. Then $\partial_{i'n-1}$ can be considered as the diameter of a circle completely enclosing the network. A is then approximated as $A = \pi\partial^2_{i'n-1}/4$. If there are q replacement nodes in the reduced network, then there will be q groups, the mean area of each being $A_g = \pi\partial^2_{i'n-1}/4q$. Thus r can be approximated as $r = \partial_{i'n-1}/2\sqrt{q}$. (since $q\pi r^2=A$) In future, these values of A and r will be taken either as available or easily approximated.

6.1.3.2.1. Estimating the Function f(x)

It is necessary to introduce the notion of clustering, although a formal definition will not be attempted. In some networks the nodes will be distributed over the area A in a very uniform pattern. This will produce the shape of Δ curve shown in Fig 6.d with reasonably smooth jumps in distance from the j^{th} nearest neighbour to $j+1^{th}$ nearest neighbour. Other arrangements will imply greater clustering tendencies such as that shown in Fig 6.e. Here the $1^{st}, 2^{nd}, .., k^{th}$ nearest neighbours tend to be closer and the Δ curve looks rather more like the one shown. The shape of this curve can be used to measure the clustering of the network in question in a number of ways. One way is to measure the point at which the curve crosses half way (see Figs 6.d and 6.e). In other words, find j' such that

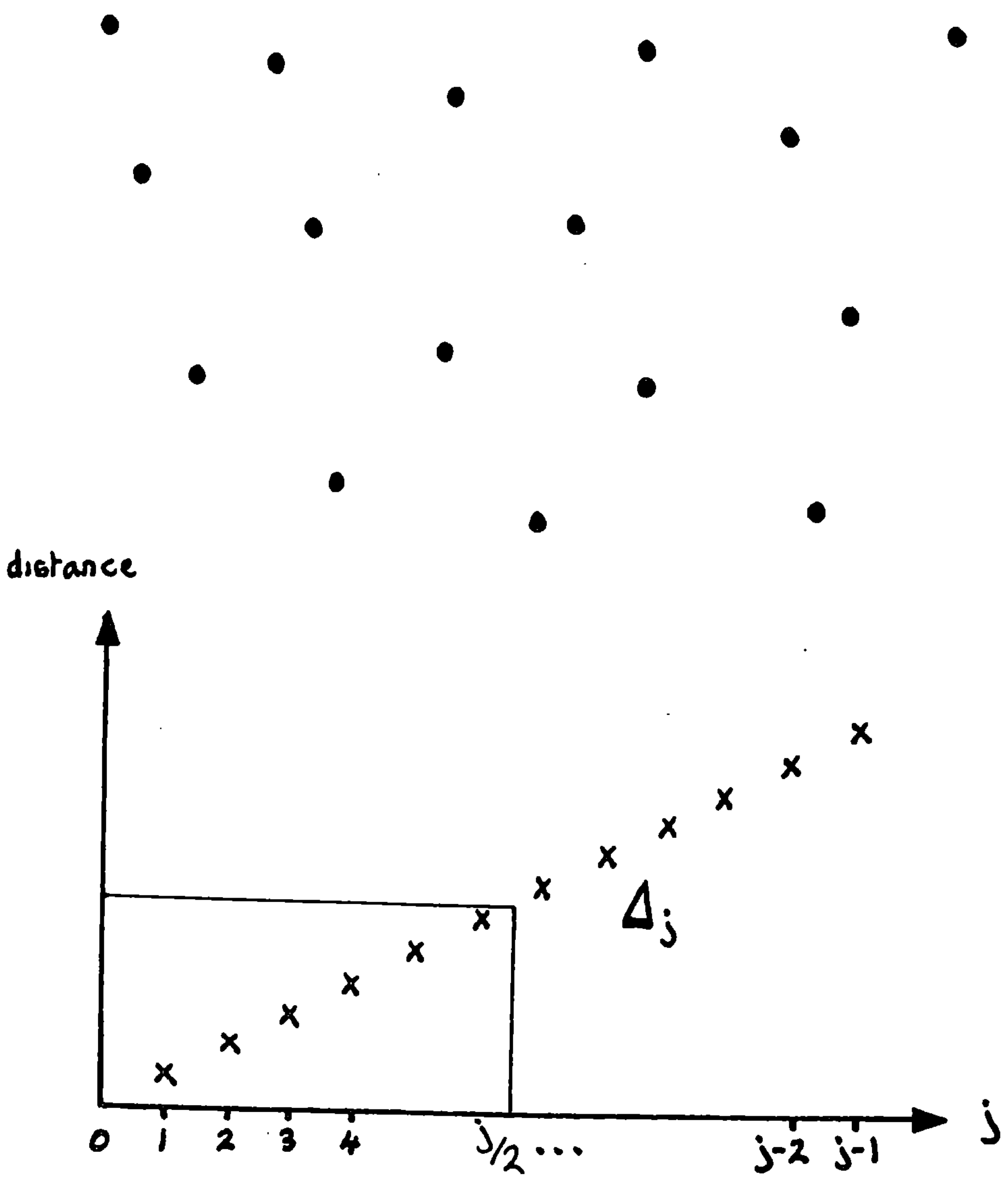
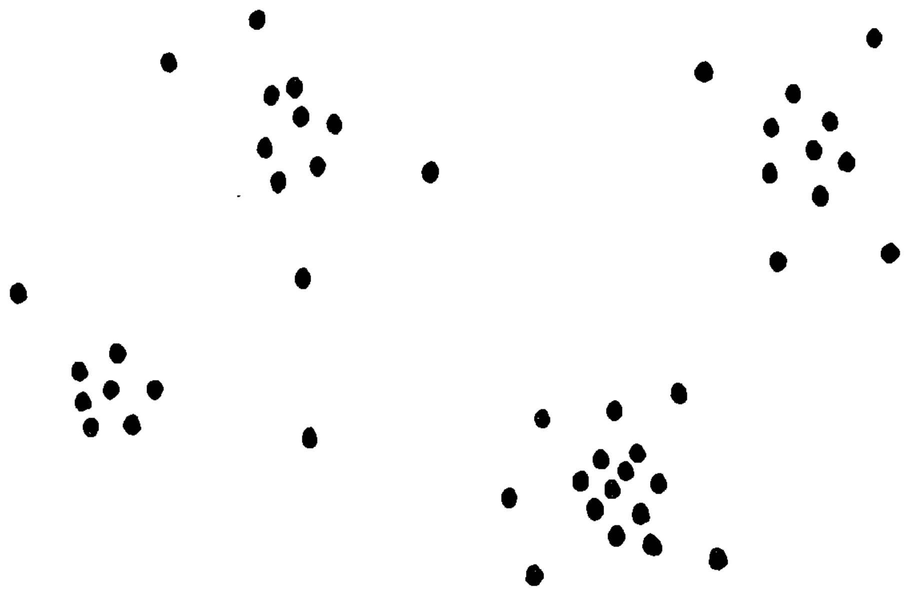


Figure 6.d Uniform Node Distribution and Δ_j Curve



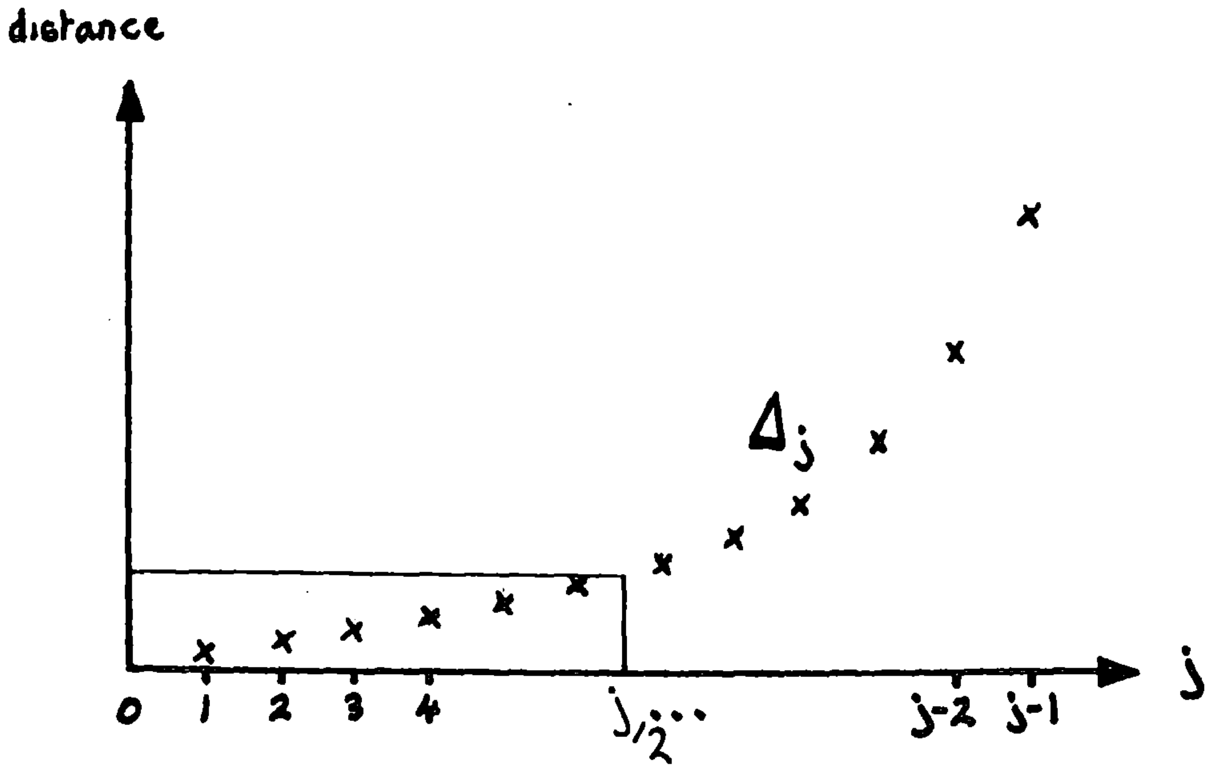


Figure 6.e. Clustered Node Distribution and Δ_j Curve

$\Delta_{j'} \geq \Delta_{n-1}/2$

The closer j' is to (n-1)/2, the more uniform will be the network distribution. The closer j' is to n-1, the more clustered the network distribution.

The extreme forms of the f(x) curve for uniform and clustered distributions are shown in Fig 6.f. A typical (partially clustered) curve is also shown. In general, the probability of a node being at a distance x from the centre of a group increases as x decreases, this feature being more pronounced in clustered networks than uniform ones.

The precise shape of the f(x) curve will depend upon the particular network in question. It can, however, be approximated in a number of ways. The exponential forms ($\alpha e^{-\alpha x}$, $\alpha e^{-\beta x}$, $\alpha e^{-\alpha x}+\beta e^{-\beta x}$, αx^{β}, etc.) can all be made to fit but, with no extra information about the network, the linear form of $f(x)=mx+c$ is as good as any. Some typical curves are shown in Fig 6.g. The extreme values of m will be 0 (uniform distribution) and $-\infty$ (extreme clustering). Clearly the distribution is only defined for $0 \leq x \leq r$ (if $x>r$ for the node in question then it is in another group). The value of c can be calculated on the basis of m (for which there are two cases to be considered) in the following way (see Fig 6.g).

1. $0 \geq m \geq -2/r^2$.

$$\int_0^r (mx+c)\,dx = 1$$

$$c = (2-mr^2)/2r$$

2. $-2/r^2 > m > -\infty$.

$$\int_0^a (mx+c)\,dx = 1 \qquad \text{where } ma+c=0$$

$$\int_0^{-c/m} (mx+c)\,dx = 1$$

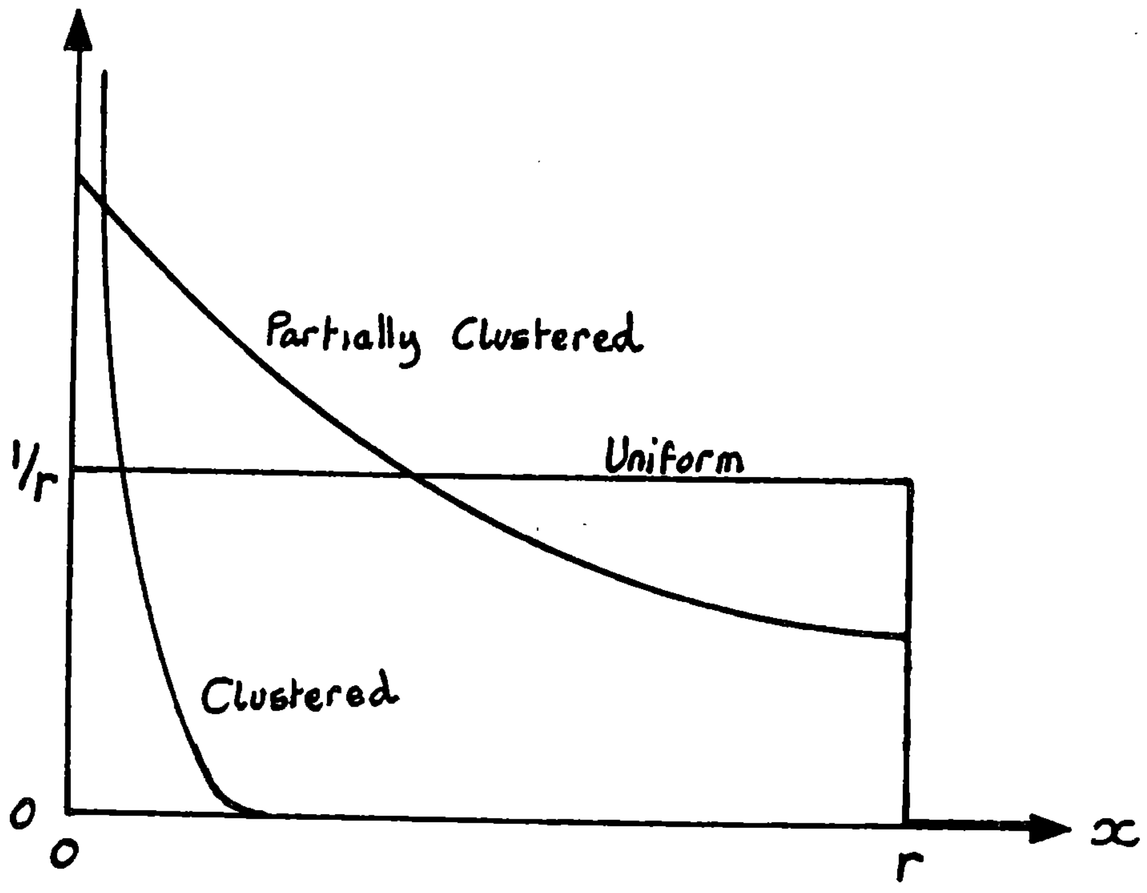


Figure 6.f. Possible f(x) Curves

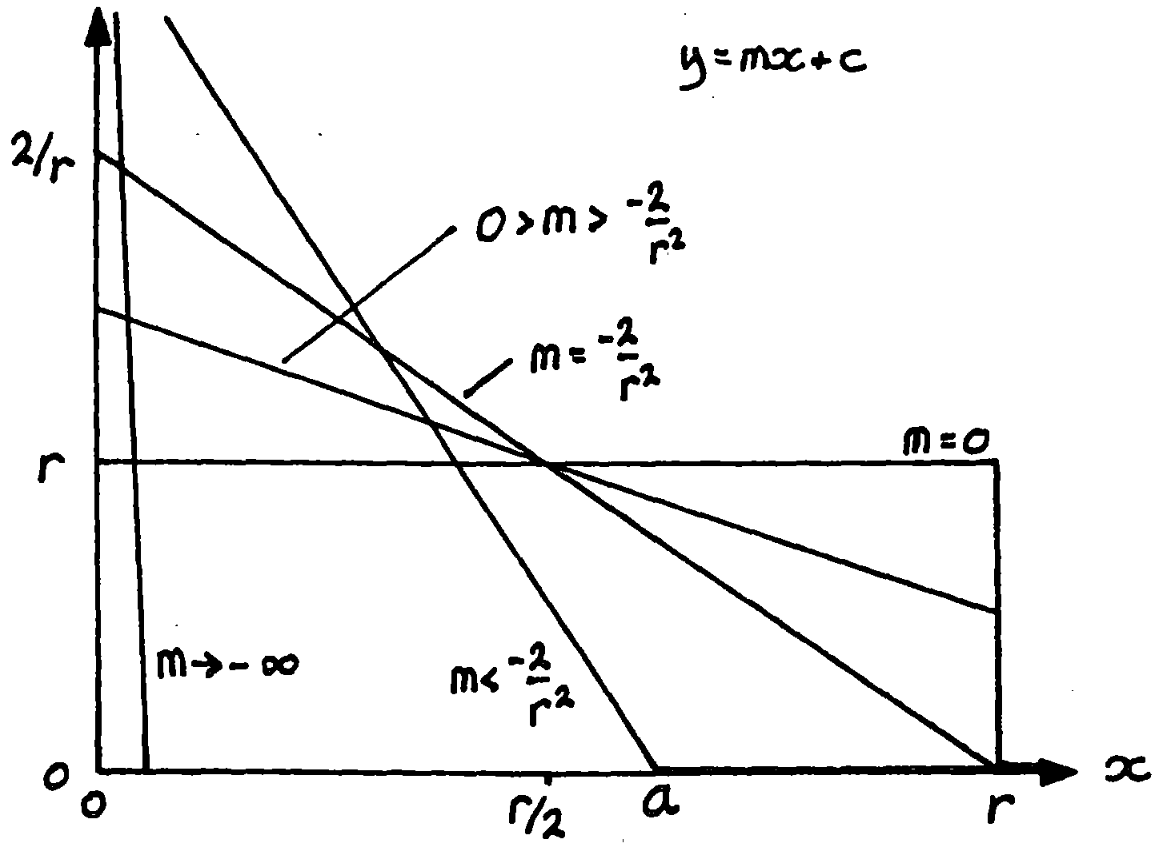


Figure 6.g. Linear Approximation of f(x)

$$c = \sqrt{[-2m]}$$

The only remaining problem is how to calculate the value of m from j' in the first place. Recall that $0 \geq m \geq -\infty$ and $(n-1)/2 < j' < n-1$. A suitable formula for m is therefore $m = -K(j' - (n-1)/2))$ where K is a large constant. K can be chosen in a number of different ways. The value bisecting $(n-1)/2$ and $n-1$ is $(3n-3)/4$ and the value bisecting 0 and r is $r/2$. Suppose that $a = r/2$ when $j' = (3n-3)/4$. Then

$$mr/2 + c = 0 \quad \text{and} \quad \int_0^{r/2} (mx+c)\, dx = 1$$

$$c = -mr/2 \qquad mr^2/8 + cr/2 = 1$$

$$-mr^2/8 = 1$$

$$m = -8/r^2$$

So $-8/r^2 = -K(n-1)/4$, implying that $K = 32/(r^2(n-1))$. Of course, K can be calculated in other ways.

As an aside, the meaning of any particular value of m can be seen by calculating the expected distance of any node from the centre of the group. For the above case with $m = -8/r^2$ and $c = 4/r$, this is given by

$$E(x) = \int_0^{r/2} (4x/r - 8x^2/r^2)\, dx = r/6$$

6.1.3.2.2. Estimating the Function $g_{n'}(y)$

$g_{n'}(y)$ is the PDF of the distance between nearest neighbours with n' nodes in the network. The curve of $g_{n'}$ will be something of the form shown in 6.h. The expected distance between nearest neighbours, $y_{n'}$, depends upon the value of n'. As n' decreases, $y_{n'}$ increases and vice versa. The actual form of the $g_{n'}$ curve is far more difficult to determine, as is the exact value of $y_{n'}$ in the general case. Both will, in fact, depend entirely upon the network distribution under

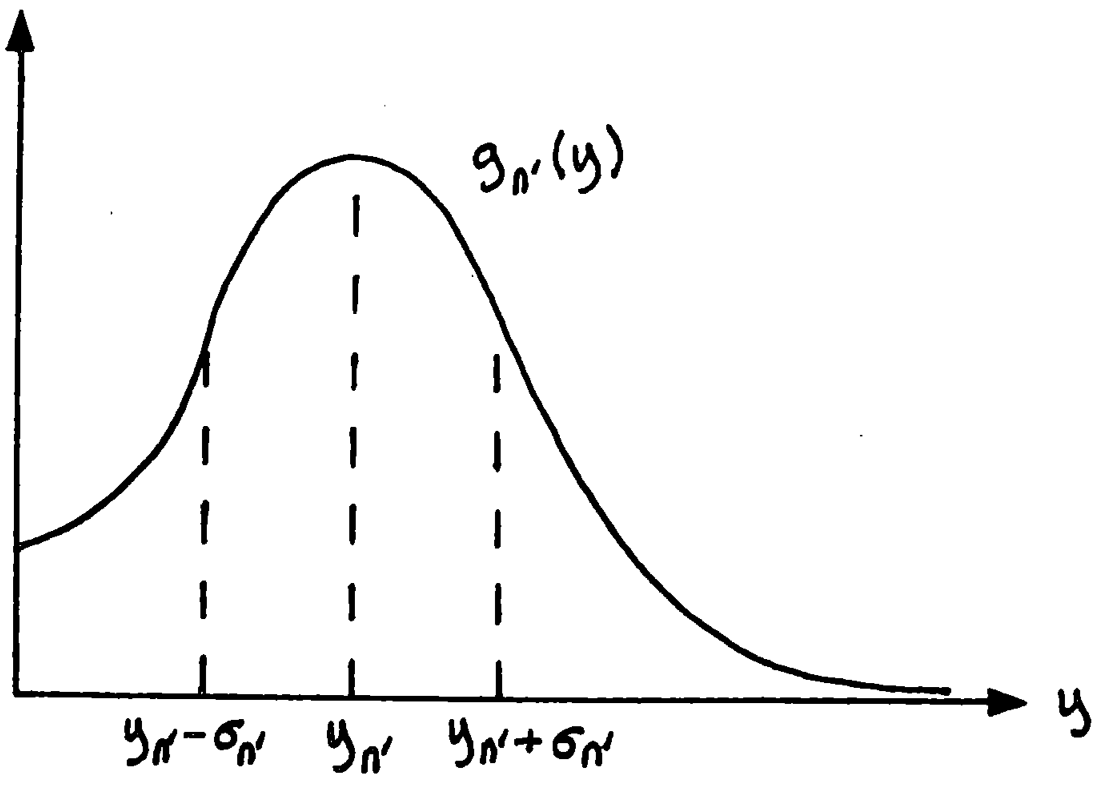


Figure 6.h. Form of the $g_{n'}(y)$ Curve

consideration. Equally difficult to calculate, but of similar importance, will be the standard deviation of the distribution, $\sigma_{n'}$.

A number of possible candidates exist for the purposes of modelling the $g_{n'}$ curve such as the Beta, Weibull, Cauchy or Lognormal distributions. These are not dealt with here - see Derman et al. [1973] for an excellent treatment.

As a brief example, consider the approximation of $g_{n'}$ by the Gamma distribution, i.e.

$$g_{n'}(y) = \begin{cases} (\theta^s y^{s-1} e^{-\theta y})/\Gamma(s) & : \ y \geq 0 \\ 0 & : \ y < 0 \end{cases}$$

where the gamma function, $\Gamma(s)$, is given by

$$\Gamma(s) = \int_0^\infty x^{s-1} e^{-x}\, dx \qquad (s>0)$$

Examples of the curve for different values of θ and s are displayed in Fig 6.1. The values of θ and s can be calculated as $s = y^2_{n'}/\sigma^2_{n'}$ and $\theta = y_{n'}/\sigma^2_{n'}$ if the mean and variance are known. s is effectively a measure of the location of the distribution along the y axis whereas θ gives an idea of the deviation from the central value. As s increases, it matches closer and closer the mean (and variance). The distance between the two nearest nodes will, generally, be smaller than the average but the system becomes more uniform as as n' decreases. The expected distance between nodes then can be taken as $s = 2\surd[A/n'\pi]$.

Even if this approach was to prove to be satisfactory (which may be in some doubt), it is extremely unlikely that it would be suitable for manipulation in the integrals forming the basis of the calculation of P and E. Some simplification is necessary.

The most appropriate, suitably convenient, approximation is one based on a triangle (see Fig 6.j). Firstly, consider a distribution of the form

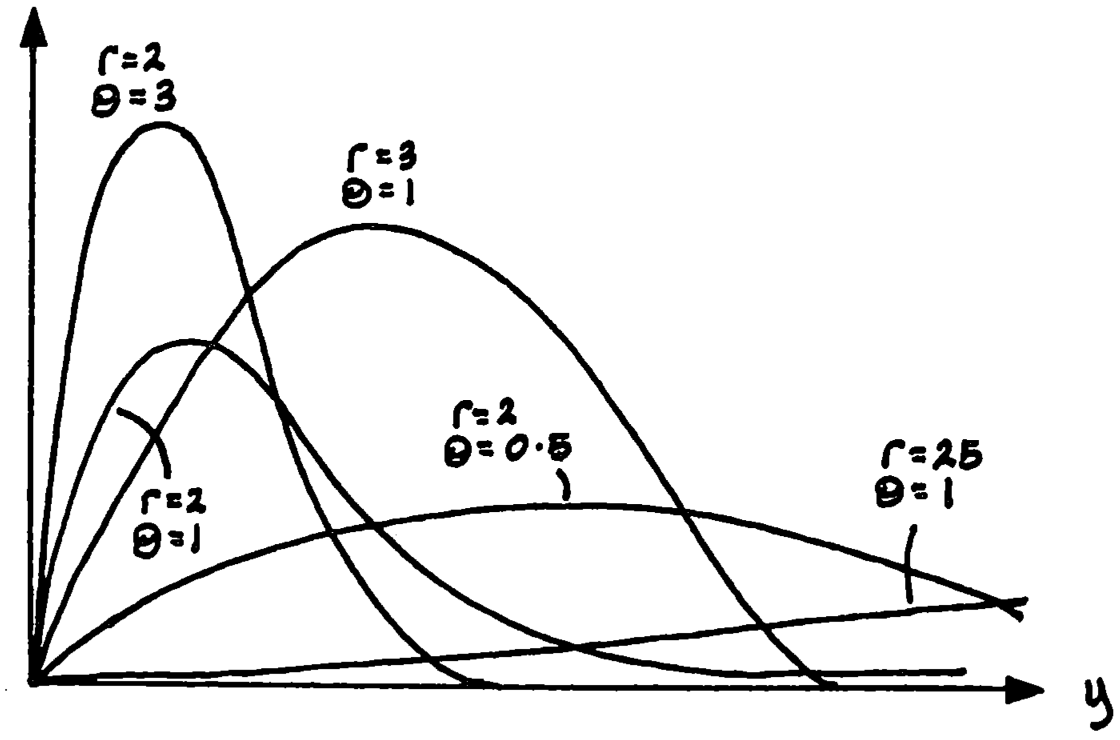


Figure 6.1. The Gamma Distribution

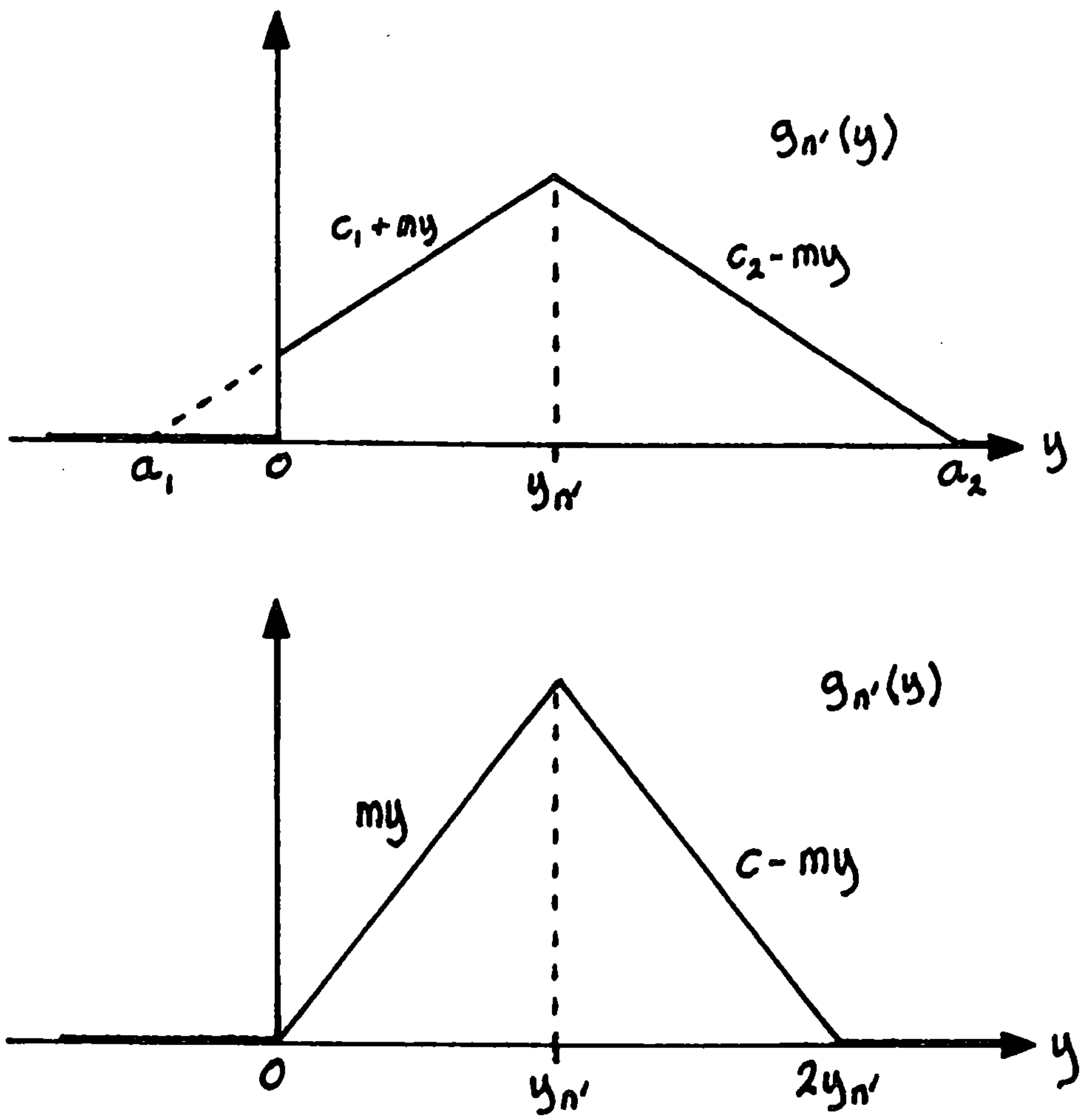


Figure 6.j. Triangle Approximation to the $g_{n'}(y)$ Curve

$$g_{n'}(y) = \begin{cases} 0 & : \quad y<\max(0,a_1) \\ c_1+my & : \quad \max(0,a_1)\le y\le y_{n'} \\ c_2-my & : \quad y_{n'}\le y\le a_2 \\ 0 & : \quad y>a_2 \end{cases}$$

where $y_{n'}=2\sqrt{[A/n'\pi]}$. Clearly, $a_1=-c_1/m$ and $a_2=c_2/m$, and the constraint $[c_1+my]_{y=yn'}=[c_2-my]_{y=yn'}$ implies that $c_1+2my_{n'}=c_2$. The fourth constraint, that the area under the curve be equal to unity, is, however, of the form

$$\int_{\max(0,a_1)}^{y_{n'}} (c_1+my)\ dy \ + \int_{y_{n'}}^{a_2} (c_2-my)\ dy \ = \ 1$$

and the imprecise nature of the limits render this unsuitable.

Consider the second triangular form in Fig 6.j. In this case, $g_{n'}$ is given by

$$g_{n'}(y) = \begin{cases} 0 & : \quad y<0 \\ my & : \quad 0\le y\le y_{n'} \\ c-my & : \quad y_{n'}\le y\le 2y_{n'} \\ 0 & : \quad y\ge 2y_{n'} \end{cases}$$

The area of each half of the triangle must be 1/2. Applying these constraints gives $m=1/y^2_{n'}$ and $c=2/y_{n'}$, giving an approximate form of $g_{n'}$ as

$$g_{n'}(y) = \begin{cases} 0 & : \quad y<0 \\ y/y^2 n' & : \quad 0\le y\le y_{n'} \\ 2/y^2 n' - y/y^2 n' & : \quad y_{n'}\le y\le 2y_{n'} \\ 0 & : \quad y\ge 2y_{n'} \end{cases}$$

6.1.3.2.3. Estimating the Function b(x,y)

The most obvious way of estimating the function b(x,y) is by considering each group as being circular of radius r. The probability that the nearest neighbour, Y, of a node, X, a distance x form the centre of the group, is outside the group, given that it is a distance y from X is equivalent to the fraction of the circle (centre (x,0), radius y) that lies outside of the circle (centre (0,0), radius r). This is illustrated in Fig 6.k.

Clearly, if $y<r-x$ then $b(x,y)=0$ whereas if $y>r+x$, $b(x,y)=1$. It remains to determine the situation where $r-x\le y\le r+x$. Using the notation of Fig 6.k, it is clear, in this case, that $b(x,y)=\theta/\pi$.

The equation of C is $a^2+b^2=r^2$ and the equation of C' is $(a-x)^2+b^2=y^2$. Eliminating b between both equations gives the a coordinate of P as $(r^2+x^2-y^2)/2x$.

Then $p = (r^2+x^2+y^2)/2x - x = (r^2-x^2-y^2)/2x$
and $\theta = \cos^{-1}[(r^2-x^2-y^2)/2xy]$

So finally, $$b(x,y) = \begin{cases} 0 & : \ y<r-x \\ 1/\pi\cos^{-1}[(r^2-x^2-y^2)/2xy] & : \ r-x\le y\le r+x \\ 1 & : \ y>r+x \end{cases}$$

It is clear that the function b(x,y), in this form, will be extremely difficult to deal with. The shape of the curve is shown in Fig 6.l. This is effectively a <u>probability distribution function</u> (PdF) between the limits r-x and r+x. A suitable approximation, in many cases, will be the linear one shown. This approximation of the function b(x,y) will be of the form

$$b(x,y) = \begin{cases} 0 & : \ y<r-x \\ (y+x-r)/2x & : \ r-x\le y\le r+x \\ 1 & : \ y>r+x \end{cases}$$

This form is likely to make the evaluation of P and E much simpler in most cases.

6.1.3.2.4. Estimating the Values of P and E

The three previous sections have produced approximations of the three component functions as follow.

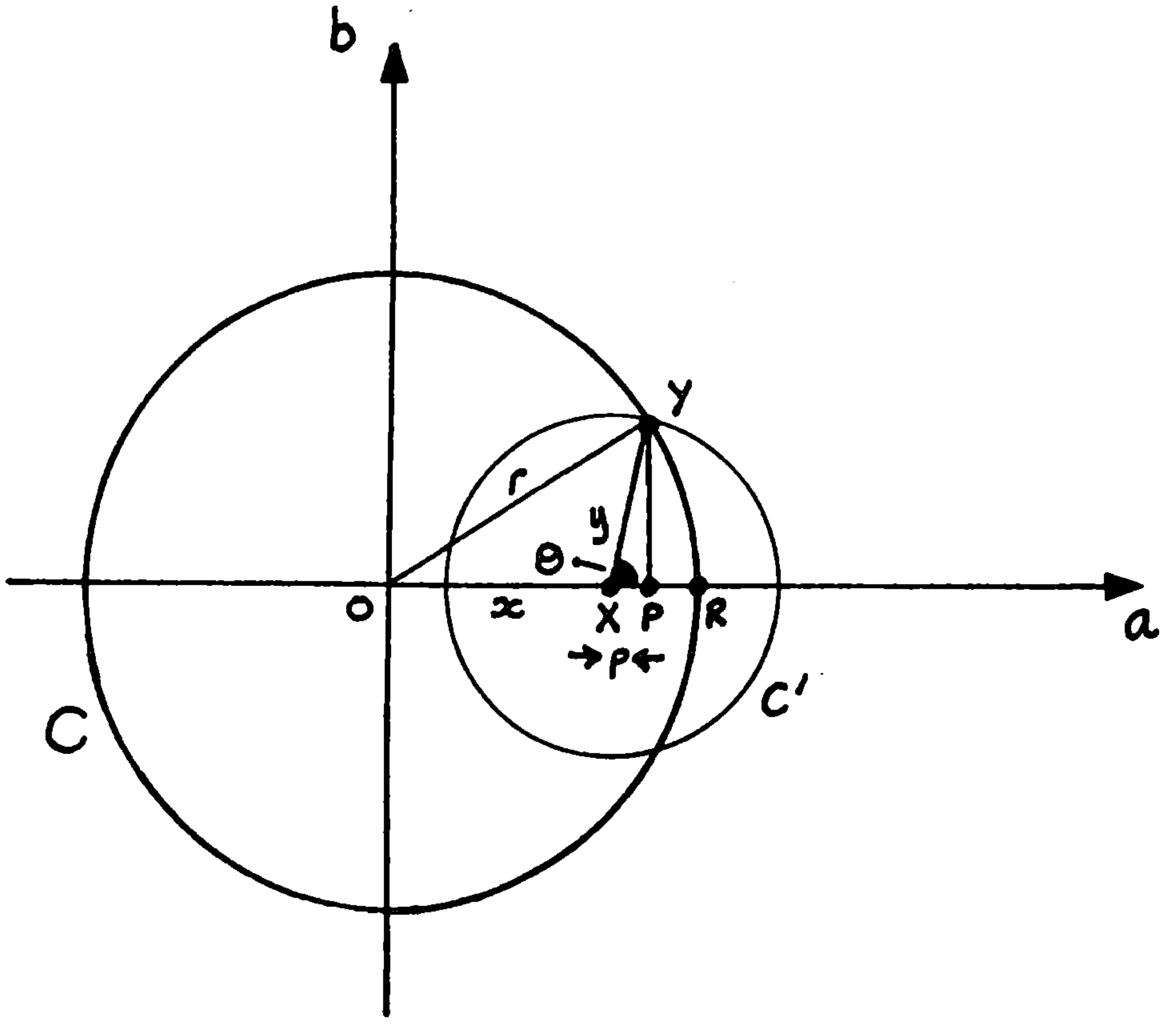


Figure 6.k. Geometrical Estimation of b(x,y)

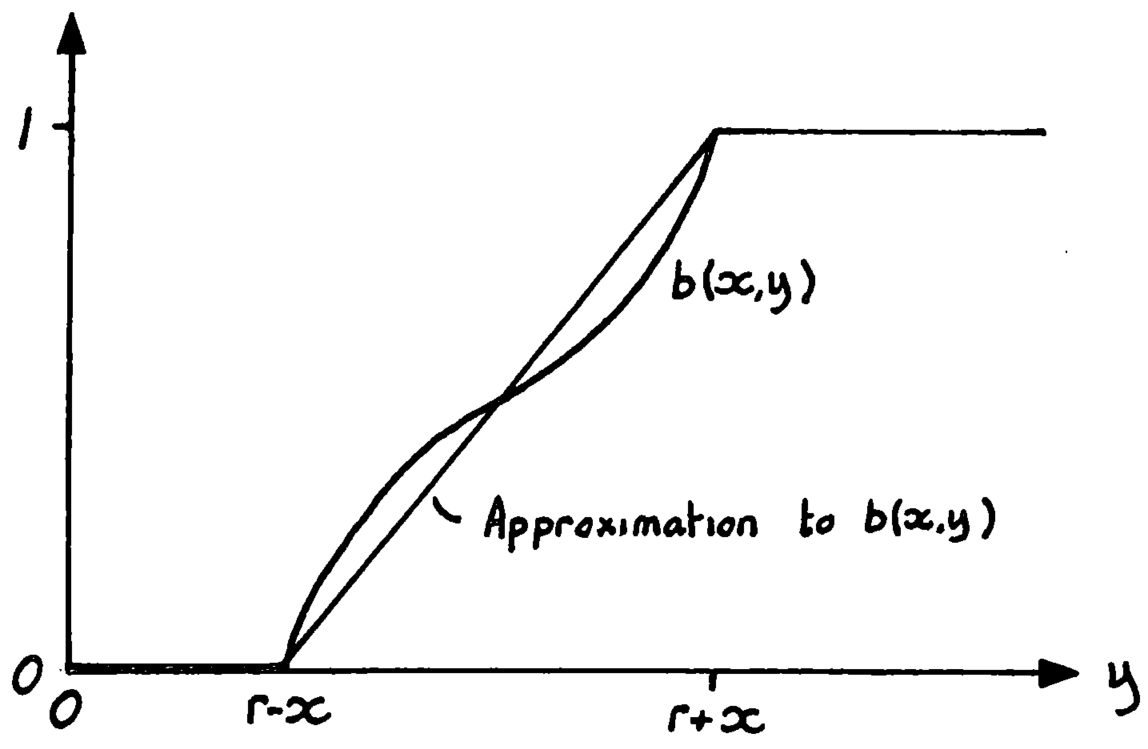


Figure 6.1. Approximation of b(x,y) Curve

$$f(x) = \begin{cases} 2/r - 2x/r^2 & : 0 \le x \le r \\ 0 & : x<0 \ x>r \end{cases} \qquad \text{(taking } m=-2/r^2\text{)}$$

$$g_{n'}(y) = \begin{cases} y/y^2_{n'} & : 0 \le y \le y_{n'} \\ 2/y_{n'} - y/y^2_{n'} & : y_{n'} \le y \le 2y_{n'} \\ 0 & : y<0 \ y>2y_{n'} \end{cases}$$

$$b(x,y) = \begin{cases} 0 & : y<r-x \\ (x+y-r)/2x & : r-x \le y \le r+x \\ 1 & : y>r+x \end{cases}$$

where $y_{n'}=2\surd[A/n'\pi]$. The expression

$$\int_0^r f(x) \int_0^\infty g_{n'}(y)b(x,y)\ dy\ dx$$

however, cannot be integrated explicitly since the order of $r-x$, $r+x$, $y_{n'}$ and $2y_{n'}$ vary in relative size. The following combinations are possible.

Case 1	:	$r-x \le r+x \le y_{n'} \le 2y_{n'}$
Case 2	:	$r-x \le y_{n'} \le r+x \le 2y_{n'}$
Case 3	:	$r-x \le y_{n'} \le 2y_{n'} \le r+x$
Case 4	:	$y_{n'} \le r-x \le r+x \le 2y_{n'}$
Case 5	:	$y_{n'} \le r-x \le 2y_{n'} \le r+x$
Case 6	:	$y_{n'} \le 2y_{n'} \le r-x \le r+x$

The integral must be evaluated individually for each of the six cases. Using the substitution $r^2=(n'y^2_{n'})/4q$ (from $A = q\pi r^2 = n'\pi(y_{n'}/2)^2$), each can be obtained in terms of n' and q. The two extremes, case 1 and case 6, for example give $P(n') = 1 - 19n'/144q$ and $P(n') = 0$ respectively. For case 1 with $q=30$ and $n'=31$, $P(n')=0.86$, suggesting that errors are likely at the latter stages.

This is a somewhat complicated, and possibly unreliable, process to pursue for each case. A simpler solution can be achieved by returning to the original definition of P(n').

$$P(n') \; = \; \int_0^r f(x)p_{n'}(x)\,dx$$

It is possible to approximate $p_{n'}(x)$ without recourse to $g_{n'}(y)$ and $b(x,y)$. A particularly suitable choice, in terms of both simplicity and flexibility, is as follows.

$$p_{n'}(x) \; = \; \begin{cases} x^h/2r^h & : 0 \le x \le r \\ 0 & : x<0 \;\; x>r \end{cases}$$

where $h \in \mathbb{R}$. This fits intuitively since $p_{n'}(0)=0$ and $p_{n'}(r)=1/2$ for all h. The precise value of h is dependent upon the network in question. It will need to be matched against practical examples (but see chapter 7 for actual $p_{n'}(x)$ curves). This section is concluded with a few examples.

Let $h=1$. Then

$$P(n') \; = \; \int_0^r (mx+c)x/2r\,dx \; = \; r(2mr+3c)/12$$

Suppose $m=-4/r^2$ (slightly clustered). Then $c=2\sqrt{2}/r$ and $P(n')=0.04$. For a network of $n=100$ nodes, reduced to one of $q=30$ replacement nodes, $E=70\text{x}0.04 = 2.8$ and $P=1-0.96^{70} = 0.94$.

Let $h=2$. Then, by a similar process, $P(n') = r(3mr+4c)/24$. Suppose $m=0$ (completely uniform). Then $c=1/r$ and $P(n')=1/6$. For $n=100$ and $q=30$, as before, $E=70\text{x}1/6 = 11.67$ and $P=1-(5/6)^{70} = 1$. (≥ 1 error).

Results of tests on practical networks are presented in chapter 7. A comparison between theoretical and practical results is also given.

6.1.3.3. Complexity of the Algorithm

The reduction algorithm consists of n-q stages. Within each stage, n', the shortest distance between pairs must be determined.

There are $n'(n'-1)/2$ such distances to be tested. The two appropriate nodes are then combined. The number of steps is therefore

$$\sum_{n'=q+1}^{n} [Cn'(n'-1)/2 + D]$$

where C is an integer constant denoting the number of steps required to calculate the distance between each node pair and D is another representing the number of steps required to make the replacement. The number of steps is thus bounded by $O(n^3)$. This, being a polynomial, satisfies the criterion for an acceptable algorithm suggested in section 3.2.

Shamos [1978] presents an algorithm which calculates the closest two points in a plane in O(nlogn) steps. This reduces the above time to $O(n^2 \log n)$.

6.1.4. Variations and Extensions

This section on the theory of network reduction concludes with a few comments about possible ways in which the method could be extended or made more flexible.

The first concerns the way in which the distance between two nodes (real or replacement) is measured. Dependent upon the application in question (and the appropriate cost functions), Euclidean distances may or may not be the most suitable.

Define a <u>metric</u> d acting upon a set S to be a distance function $d:S \times S \to \mathbb{R}$, satisfying the following conditions.

M1. $d(a,b) \geq 0$ for all $a,b \in S$ with equality iff $a=b$

M2. $d(a,b)=d(b,a)$ for all $a,b \in S$

M3. $d(a,c) \leq d(a,b)+d(b,c)$ for all $a,b,c \in S$

Even when S is restricted to be a set of coordinates $\{(x_i, y_i)\}$, the Euclidean metric, denoted d_2, is still but one possibility. Others are $d_1(a,b):=|x_a-x_b|+|y_a-y_b|$ and $d_\infty(a,b):=\max(|x_a-x_b|, |y_a-y_b|)$ which are special cases of the general form $d_k(a,b):=\sqrt[k]{[(x_a-x_b)^k+(y_a-y_b)^k]}$. The closet pair, in the most general case can be selected on the basis of the most appropriate metric.

There are two main reasons why the choice of metric may vary. Firstly, the cost of links may be calculated in a distinctly nonlinear fashion and secondly, connections may not be permitted to follow the direct route between nodes (e.g. the Manhattan form of distance for which the d_1 metric applies).

Another generalisation would be to replace more than two nodes at a time. This leads on to the concept of a <u>multidimensional metric</u> $d: S^v \rightarrow \mathbb{R}$ satisfying

MDM1. $d(\underline{a}) \geq 0$ for all $\underline{a} \in S^v$ with equality iff $a_i = a_j$ for all $i, j = 1, 2, .., v$

MDM2. $d(\underline{a}) = d(\underline{a}^*)$ for all permutations $\underline{a}^*$ of the elements of $\underline{a} \in S^v$

MDM3. $d(a,b,..,i,..,k,..,z) \leq$
$d(a,b,..,i,..,j,..,z) + d(a,b,..,j,..,k,..,z)$ for all vectors in S^v

Applying the appropriate multidimensional metric will allow the closest group of v nodes to be replaced by a single one. The vector $\underline{a}^*$ is found such that $d(\underline{a}^*) = \min_{a \in S} v(d(\underline{a}))$ and the replacement node details are given by

$$x_r = \frac{\sum_{i=1}^{v} w_i w_j}{\sum_{i=1}^{v} w_i} \qquad y_r = \frac{\sum_{i=1}^{v} w_i y_i}{\sum_{i=1}^{v} w_i} \qquad w_r = \sum_{i=1}^{v} w_i$$

Care must be taken at the end of the process to ensure that the value of q is not leapfrogged. The value of v may need to be decreased for the last stage.

A possible alternative (though not one which is examined in this work) would be to maintain the concept of replacing nodes in pairs but to replace more than one pair at a time (e pairs, say). This does, however present a couple of problems in that sorting techniques will be required which could increase the complexity and there might be some confusion in the situation where the same node found itself included more than once in the closest e pairings.

As mentioned previously, the weights of the nodes can be determined in a number of ways. Similarly, the replacement operations $x_r := \dots$, $y_r := \dots$ and $w_r := \dots$ can be varied. Such concepts, however, appear to be distanced from the problems concerned with telecommunication networks and are not considered in any detail here. Some possible extensions of the reduction method to other areas are considered in chapter 8.

6.2. Fixed Topology Optimisation

This section is concerned with the determination of the optimum tandem set or an approximation to it. The problem is to select Ω tandems from the n original nodes (and to determine the value of Ω). The n initial nodes may, or may not, have been replaced by q representative ones using a reduction method presented in the previous section. It will be assumed, for the purposes of this section, that the n original nodes *have* been replaced by q nodes in this way. If this is not actually the case then simply set q=n and proceed as if replacement had taken place. The notation $I:=\mathcal{N}'$ will be used, at the beginning of any algorithm, to indicate that data concerning a (possibly) reduced network is input. Normally, $\mathcal{N}'$ will have been obtained from $\mathcal{N}$ by the techniques comprising the reduction process (a reduced number of node locations, smaller traffic matrices, etc.). If no reduction has taken place, however, $\mathcal{N}'=\mathcal{N}$.

The q nodes will be taken as being actual sites (i.e. it will be assumed that each replacement node has been moved to its nearest actual site if necessary). Under any other circumstances, a similar operation can be performed on the greenfield tandems which are eventually chosen. The methods presented in this section are applicable in either case.

There are three possible directions of approach in the search for tandem sets. In order of decreasing complexity, these are as follows.

1. Exact optimisation methods
2. Accurate heuristic methods
3. Fast heuristic methods

In principle, at least, the reduction process of section 6.1 could be continued until q is small enough to allow exact methods to operate. In practice, however, the value of q which would permit such an approach is too small for realistic optimisation (in fact $q<\Omega$ under extreme circumstances which is ridiculous - for the reduction process to be of any use at all, it must not reduce further than the eventual optimum number of tandems). Furthermore, the vast amount of extra work incurred by exact methods rarely justifies the small (often nonexistent) improvement in the solution.

The methods presented here, and for the rest of the chapter, are of type 2. The value of q is taken to be such that accurate heuristic techniques are applied which could not be expected to operate on the initial network due to its size. It transpires that these techniques, acting upon the reduced network, will produce results superior to those obtained from methods of type 3 working with the original.

An accurate heuristic method for determining the tandem set is presented in this chapter, based upon a temporary restriction of the valid solution space.

6.2.1. Principles of Fixed Topologies

The difficulties involved in selecting the optimum tandem set are familiar. With q nodes and Ω tandems, there are $q!/[\Omega!(q-\Omega)!]$ tandem sets, from which the best is to be chosen (or approximated). For each set of M tandems ($M=1,2,..,\Omega,..,q$) there are over $2^{M(M-1)/2-1}$ valid connection topologies. For all but trivially small values of q, this number is prohibitively large. It is impractical to generate and evaluate each tandem set and connection topology in turn.

The approach suggested in this section is one in which the connection topology solution space is restricted. The number of tandem sets is manageable, provided that the number of connection topologies, for each one, is kept small. The most natural way to achieve this, rather than arbitrarily dismissing certain forms, is to specify a particular connection topology to be used in all cases.

There are various ways in which this restriction can be made. A number of distinctive connection topologies exist which can be applied to a set of M tandems relating, in essence, to the named graphs of chapter 2. Any one of these can be chosen and its form demanded for the tandem network. It is vital to realise that this restriction is not permanent. It only applies during the course of the selection of the tandems. Other, improved,methods will be used to determine the tandem network connection topology once these tandems are established. In fact, it will be seen that even the positions of the tandems are not fixed permanently yet, but that this is the purpose of the algorithms suggested in section 6.3.

A method of tandem set determination is suggested based on the temporary assumption of a star-shaped tandem network.

6.2.2. Star Optimisation

There is a strong tendency for practical tandem networks to have structures similar to stars as shown in Fig 6.m. This is not an accident. The requirements of traffic collection and distribution are particularly well satisfied by this form of hierarchical centralised design. Trees, in general, minimise connection distance and a star is merely a special type of tree allowing strict DOR. In many practical cases, the first circuit in a link will cost comparatively more than subsequent links. It is therefore often advantageous to reroute traffic over a slightly longer distance distance in order to dispense with a link altogether. Whilst a star topology will not be optimal in all cases, it provides by far the most suitable fixed topology to use

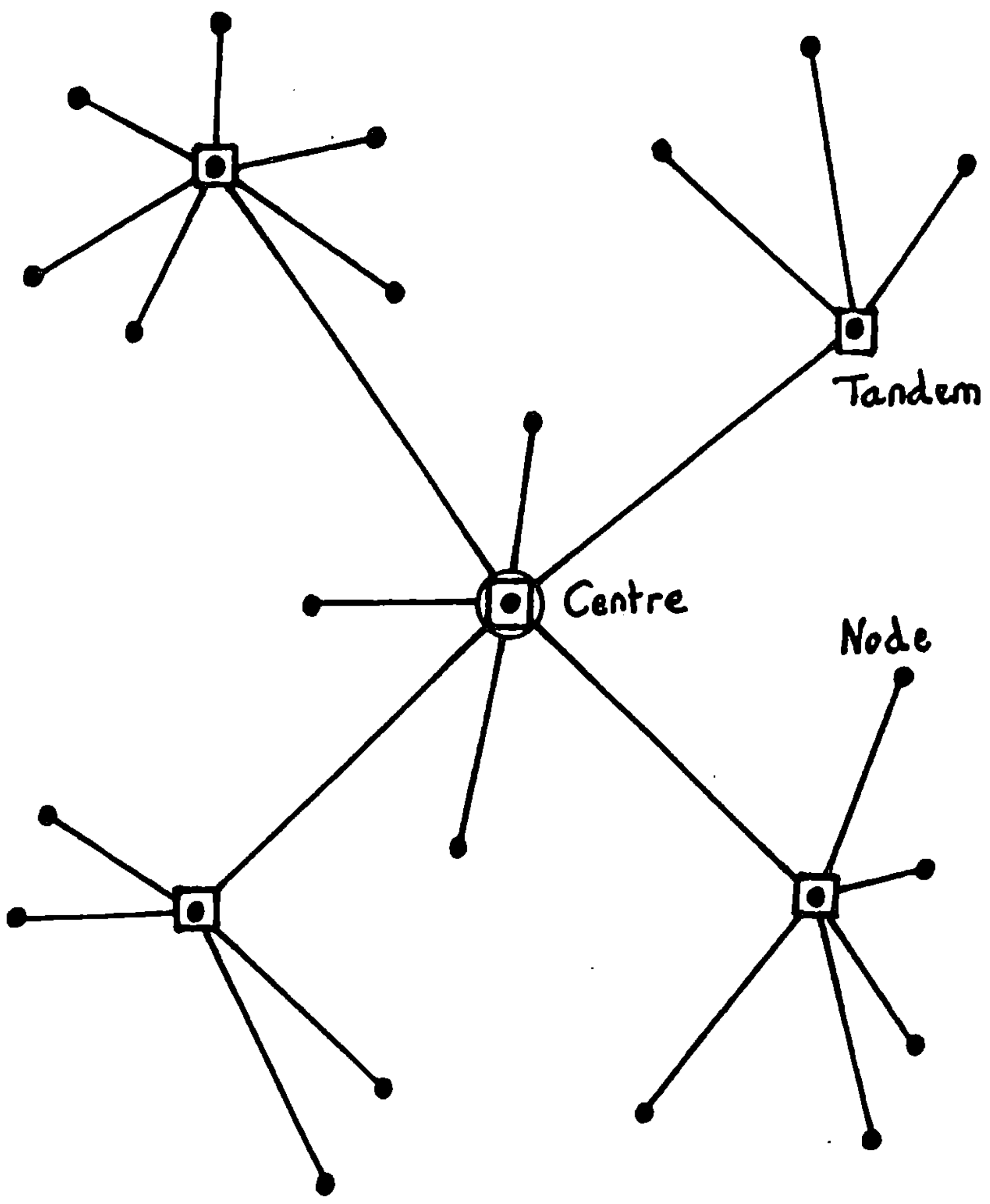


Figure 6.m. Star Tandem Network

in the sort of approach suggested here. In addition, the central tandem may equate to a head office or some other point of heavy traffic.

6.2.2.1. Using a Star Tandem Network

There are only M star tandem networks which can be constructed on M tandems (each one has a different tandem as the centre of the star) in comparison with over $2^{M(M-1)/2-1}$ networks in all. If the connection topology is temporarily restricted to a star then there are, in total,

$$\frac{Mq!}{M!(q-M)!} = \frac{q!}{(M-1)!(q-M)!}$$

allowable choices of tandem sets and tandem networks. M, however, may range from 1 to q so that the number of networks to be tested, in principle, is

$$\sum_{M=1}^{q} \frac{q!}{(M-1)!(q-M)!}$$

which is, unfortunately, still too large. The trouble is no longer the enormous number of connection topologies, but the extremely large number of tandem sets. The summand becomes rapidly larger as M increases. Returning to the practicalities of the problem, however, allows a solution to be realised.

There are basically three components to the total cost of any two-level hierarchical switching network (bearing in mind that the cost of the nodes themselves is constant and therefore neglectable - see section 5.1.3). These are as follows.

1. The cost of connecting the nodes to the tandems (node-tandem links). These costs generally decrease as the number of tandems increases since there are fewer nodes to be connected

up to the tandem network.

2. The cost of interconnecting them together (tandem-tandem links). These costs generally increase as the number of tandems increases since there are more tandems to be connected together.

3. The cost of the tandem switches themselves. These costs increase smoothly as the number of tandems increases.

The combined effect of these three component costs is shown in Fig 6.n. The total cost function is the discrete equivalent of a convex (downwards) curve. One of the implications of this is that if $C^*(M)$ is the cost of the optimum network with M tandems then there exists an M^* such that

$$C^*(1) \geq C^*(2) \geq \dots \geq C^*(M^*-1) \geq C^*(M^*) \leq C^*(M^*+1) \leq \dots \leq C^*(q)$$

If this is the case then clearly $M^*=\Omega$. Far more important, however, is the uniqueness of this minimum, both locally and globally. It allows a crucial simplification to be made to the optimisation process. It is only necessary to test M=1, 2, 3, etc. as long as the cost, $C^*(M)$ is decreasing. As soon as an M' is found such that $C^*(M') > C^*(M'-1)$ then the process can terminate with $\Omega=M'-1$ and the required tandem set will be the set obtained at stage M'-1. The number of tandem networks will be thus restricted to

$$\sum_{M=1}^{\Omega+1} \frac{q!}{(M-1)!(q-M)!}$$

which is reasonable if Ω is not too large (see results in chapter 7). The selection algorithm is then as follows.

```
I:=Ø' ;
M := 0 ;
```

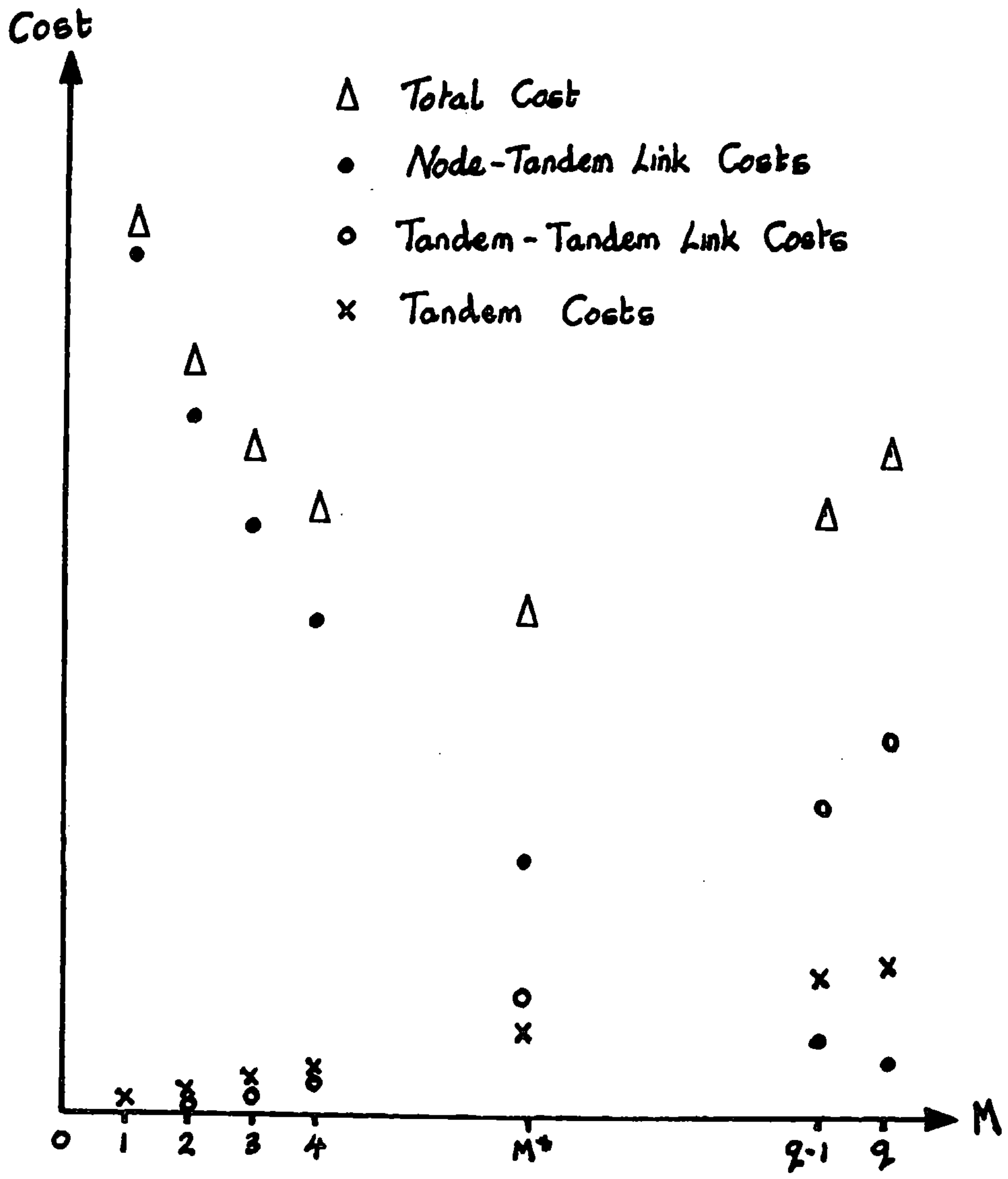


Figure 6.n. Cost Variation with Number of Tandems

```
REPEAT BEGIN
    M := M+1 ;
    $C^*(M)$ := $\infty$ ;
    generate_first_set_of_tandems ;
    REPEAT BEGIN
        FOR i := 1 TO M DO
            BEGIN
            choose tandem i as centre of star network ;
            connect all other tandems to tandem i ;
            connect each node to its nearest tandem ;
            calculate cost_of_network ;
            IF cost_of_network<$C^*(M)$ THEN
                BEGIN
                $C^*(M)$ := cost_of_network ;
                record current set of tandems ;
                END ;
            END ;
        IF there_are_more_tandem_sets_to_test THEN
            generate_next_tandem_set ;
        UNTIL there_are_no_further_tandem_sets_to_test ;
    UNTIL $C^*(M)>C^*(M-1)$ ;
$\Omega$ := M-1 ;
O={$\Omega$, selection of $\Omega$ tandems from q nodes} ;
```

To implement this algorithm, the cost of a particular network must be determined as detailed in section 5.1.3. It also requires the tandem set generating procedure discussed in section 5.2.2.1 and provided in appendix A.

Another huge advantage of the star tandem network assumption is now evident. No consideration needs to be given to traffic routeing. There is no room for flexibility. Traffic from one non-central tandem to another travels from the first tandem to the centre and, from there, to the second without alternative. Traffic to or from the centre flows along the appropriate direct link. This simplifies the problem considerably. For moderate values of q (about 20 to 60, depending upon the value of Ω - see section 6.2.4.1 and the results in chapter 7), the above algorithm is quite acceptable in terms of complexity.

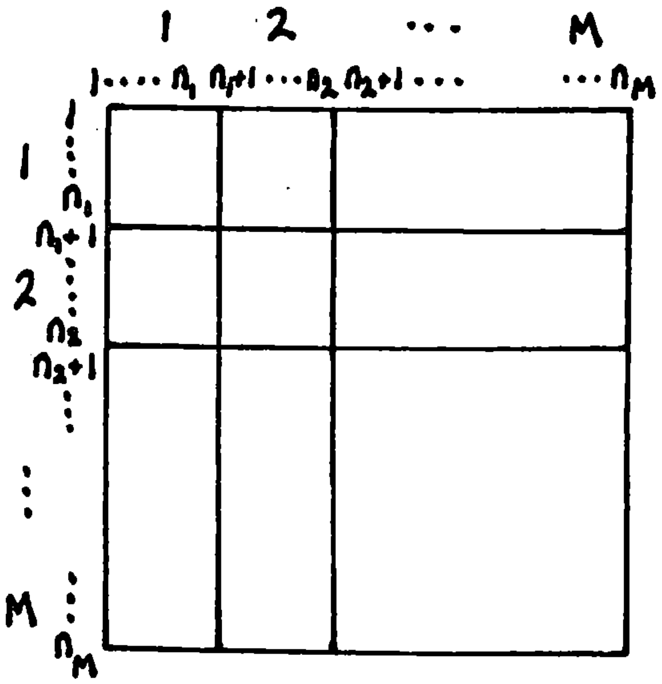
6.2.2.2. Dealing with Traffic

Suppose there are M tandems in a given configuration with the centre labelled 1 and the rest labelled 2,3,..,M. In order to calculate the cost of the star tandem network in each case it will be necessary to know the amount of traffic flowing in each arm of the star. This is clearly equivalent to the total amount of traffic A_i which each tandem i, i=2,3,..,M needs to send to the remaining tandems j=1,2,..,j-1,j+1,..,M. Calculating this traffic presents an interesting problem.

Suppose, for convenience, that the q nodes are relabelled such that nodes 1,2,..,n_1 are in tandem group 1, nodes n_1+1,n_1+2,..,n_2 are in tandem group 2, and so on with nodes n_{M-1}+1,n_{M-1}+2,..,n_M (n_M=q) in tandem group M. Again, purely for the purposes of explanation, let there be only one traffic matrix A as shown in Fig 6.o. For each tandem i (i>1) the traffic A_i is given by $A_i=A_{i1}+A_{i2}+A_{i3}+A_{i4}$ and the traffic which remains purely local to the group is A_{ii} where A_{ij}, j∈{1,2,3,4,i}, is the sum of the elements (a_{ij}) which lie in the appropriate region of the matrix. Let A_T be the total amount of traffic flowing in the network.

An obvious way of calculating A_i is to simply sum over all the appropriate elements in the q x q matrix (i.e. those elements in regions A_{i1}, A_{i2}, A_{i3} and A_{i4}). These regions, however, will vary each time a different set of transits is chosen. Each time this happens, the new regions have to be determined (using an procedure similar to that in section 6.1.2.4) and a large number of additions performed in order to calculate A_i for each tandem i. Recalling that there are q!/[M!(q-M)!] tandem sets, it is clear that such an approach will take a considerable time.

A much quicker value can be obtained heuristically. Before any of the tandem sets are generated, define α_j and β_j to be such that



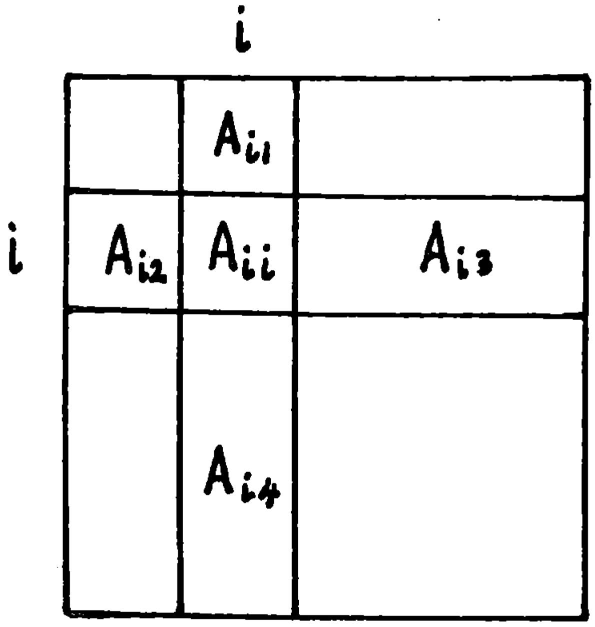


Figure 6.0. The Traffic Matrix and Local Traffic Factor

$$\alpha_j = \sum_{k=1}^{q} a_{jk} \quad \text{and} \quad \beta_j = \sum_{k=1}^{q} a_{kj} \qquad j=1,2,\ldots,q$$

α_j represents the total output traffic from node j and β_j the total input traffic. The total traffic into and out of node j is thus given by $t_j{:}=\alpha_j+\beta_j-a_{jj}$ (since a_{jj} is counted once in both α_j and β_j - a_{jj} may be non-zero, since node j may now represent a number of original nodes, each sending traffic to each other and this corresponds to traffic from node j to itself). The total traffic handled by a tandem i is therefore given by

$$A_I = \sum_{j=n_{i-1}+1}^{n_i} t_j$$

and the required traffic in the link (i,i) is given by $A_i{:}=A_I-A_{ii}$. The problem is now to find A_{ii}, the traffic which remains local to the tandem group. An exact answer is unobtainable without recourse to the original matrix (which is what this approach is seeking to avoid). Two methods of approximating A_{ii} (and hence A_i) are presented here.

The simplest approximation involves calculating $A_{ii}{:}=A_I/M$, implying that $A_i{:}=(M-1)A_I/M$ (i.e. traffic to and from tandem i is equally distributed around the network, including that which remains local to the group). This does, however, assume that all tandem groups are of the same size (in terms of traffic flow). Such a method will suffer in situations in which a large variation in tandem group sizes is in evidence. A much more acceptable solution would be one in which the varying sizes of the tandem groups is taken into account.

Define the <u>traffic density</u> D(R) of a region R of the traffic matrix to be the average element value within the region R. In other words, D(R) is given by

$$D(R) = 1/|R| \sum_{(i,j)\in R} a_{ij}$$

If the traffic flow to and from nodes in the tandem group i is reasonably balanced (which is likely for any group containing more than two or three nodes) then the ratio of traffic densities of the areas representing the traffic remaining local to the group i and the total traffic handled by tandem i should be approximately equal to the ratio of traffic densities of the areas representing the total traffic handled by tandem i and the total traffic in the network. In simpler terms, $D(A_{ii})/D(A_I) = D(A_I)/D(A_T)$. By comparison of the relative areas occupied by A_{ii}, A_I and A_T, this can be seen to imply that

$$A_{ii}/(A_I+A_{ii}) = (A_I+A_{ii})/4A_T$$

$$4A_TA_{ii} = (A_I+A_{ii})^2$$

$$A^2_{ii}+(2A_I-4A_T)A_{ii}+A^2_I = 0$$

$$A_{ii} = 2A_T-A_I+/-2\surd[A_T(A_T-A_I)]$$

Clearly $2A_T-A_I>A_T$ and $A_{ii}<A_T$ so , by inspection, the negative root is taken so that

$$A_{ii} = 2A_T-A_I-2\surd[A_T(A_T-A_I)]$$

and $A_i = A_I-A_{ii}$

$$= 2(A_I-A_T+\surd[A_T(A_T-A_I)])$$

providing a reasonably accurate approximation to the amount of traffic flowing over the link (i,i).

As a simple example, consider a situation with 5 tandems and traffic figures of

i	A_I	i	A_I
1	50	4	20
2	10	5	20
3	25		

Then A_T=125. To calculate the traffic on the link (1,3) using the simple formula $A_1=(M-1)A_T/M$ gives t(1,3)=4x25/5 = 20 whereas the improved method gives t(1,3) = 2(25-125+$\sqrt{}$[125(125-25]) = 2($\sqrt{}$12500-100) = 2(111.8-100) = 23.6.

6.2.3. Alternatives to the Star Tandem Network

Obviously a star is not the only choice for a fixed topology optimisation algorithm acting on M tandems. Any well defined graphical form will be acceptable. A loop (a connected graph in which every node - tandem in this case - has degree two) is a viable alternative, as is a wheel (a graph formed by connecting every node of a M-1 node loop to an extra node). As mentioned, however, stars are more practical. Two alternatives to such a topology are discussed in the next two sections. The relative strengths and weaknesses of each are considered.

6.2.3.1. Minimal Spanning Trees

Superficially, it would appear that a tandem network in the form of a MST would match a star in accurate representation of practical cases (a star is only a special case of a tree in which the length of the longest path is two links). In fact, there is only one MST for each group of M tandems as opposed to M stars so this could make testing quicker. In addition, the routeing strategy is trivial, as it is for the star, since one of the properties of a tree is that only one path exists between each pair of nodes.

There are, however, some serious problems with the MST approach. For example, a strict DOR may be applied to the solution network. A MST with long paths could violate this constraint. A set of tandems would then be produced based on a connection topology that would not be permitted in the final solution. This set could therefore be entirely inappropriate.

A far more serious problem, however, is the difficulties encountered in actually determining the MST. The simple algorithms presented in chapter 2 require that the cost of each link be constant and known in advance. The size of a link in the tandem network depends upon the amount of traffic it has to carry, which in turn depends upon the arrangement of links in its vicinity. The choice of links in the network, however, is the required output from an MST algorithm. The necessary input is therefore not available.

The alternative is to search through each of the M^{M-2} trees which can be constructed on M tandems. This, of course, compares extremely unfavourably with the M stars.

6.2.3.2. Full Mesh

A full mesh tandem network is one in which each tandem is connected to every other tandem (i.e. a complete graph on the M tandems). It is, in many ways, at the other extreme from a star or MST. All traffic flows along the direct link between the appropriate tandem pair.

There is clearly only one full mesh which can be constructed upon each set of M tandems and it is extremely easy to produce. The traffic along each link can be approximated using techniques similar to those discussed in section 6.2.2.2. If A_i and A_j are the approximate traffic flows in the links (i,i) and (j,i) of the star respectively, then the traffic along the link (i,j) in the full mesh can be approximated as $(A_i+A_j)/2(M-1)$.

A full mesh is a useful alternative to a star tandem network in a fixed topology method, particularly when the traffic levels between tandem groups are very high or, alternatively, when link costs are particularly low in comparison with switching costs. For typical networks, however, such flows do not commonly occur and the star is to be preferred in most cases.

In most of the work which follows in sections 6.3 and 6.4, the tandem set may generally be assumed to have been determined using either a star or full mesh tandem network. In general, however, a star network will be stated.

6.2.4. Discussion of Fixed Topology Techniques

The concept of temporarily adopting a certain form of tandem connection network is a useful one in quickly obtaining a set of tandems. It should not, however, ever be considered as a method of determining the final topology. Techniques for this are presented in section 6.4. This final part of section 6.2 deals with the practical application of the fixed topology method and possible sources of error before leading into section 6.3 by considering the star network as an initial solution to a step by step refinement algorithm.

6.2.4.1. Implementation

For any chosen fixed topology (star, full mesh, etc.) there will be a certain number of networks, T_M, which can be constructed on a given set of M tandems. For example, with the star, $T_M = M$ while for the full mesh, $T_M = 1$. The total number of acceptable restricted tandem networks is then given by $T_M \, {}^qC_M$. For the restriction to be of use, T_M should be no more than polynomial in M. If this is the case then ${}^qC_M$, the number of different sets of M tandems from q nodes, is the dominant factor in this expression. The following table gives the magnitude of ${}^qC_M$ for the first few relevant values of q and M by displaying b where ${}^qC_M = a \times 10^b$, $1 \le a < 10$.

q \ M	1	2	3	4	5	6	7	8	9	10
15	1	2	2	3	3	3	3	3	3	3
16	1	2	2	3	3	3	4	4	4	3
17	1	2	2	3	3	4	4	4	4	4
18	1	2	2	3	3	4	4	4	4	4
19	1	2	2	3	4	4	4	4	4	4
20	1	2	3	3	4	4	4	5	5	5
21	1	2	3	3	4	4	5	5	5	5
22	1	2	3	3	4	4	5	5	5	5
23	1	2	3	3	4	5	5	5	5	6
24	1	2	3	4	4	5	5	5	6	6

25	1	2	3	4	4	5	5	6	6	6
26	1	2	3	4	4	5	5	6	6	6
27	1	2	3	4	4	5	5	6	6	6
28	1	2	3	4	4	5	6	6	6	7
29	1	2	3	4	5	5	6	6	7	7
30	1	2	3	4	5	5	6	6	7	7

The value of b which is acceptable in a given situation will depend upon the efficiency of the processing resources available. A typical value for a small computer would be b = 4, 5 or 6. Suppose, in a particular case, b=4. For each value of M, a value q_M can be determined which represents the maximum value of q which can be used together with M without exceeding b. For the case b=4, these values are as follows.

M	q_M	M	q_M
1	>>30	6	22
2	>>30	7	20
3	>30	8	19
4	>30	9	19
5	28	10	19

The values q_M not defined exactly in this table can be worked out by more accurate investigation or approximated ($q_1=q_2=50$, $q_3=q_4=30$, for example).

This apparently presents a problem. The value of Ω is unknown at the beginning of the optimisation process. Hence the maximum value of M (i.e. Ω+1) which will need to be tested is not known either. The appropriate value of q to reduce down to at the reduction stage is therefore impossible to predict. This difficulty can be overcome using an algorithm which alternates between reduction and fixed topology tandem selection.

RTS1. Set M=0, $C^*(0)=\infty$ and $q_0=n$.

RTS2. M=M+1. Find q_M from tables.

RTS3. Reduce the current network from q_{M-1} nodes to q_M nodes.

RTS4. Use a fixed topology procedure to find the best set of M tandems. Calculate $C^*(M)$.

RTS5. If $C^*(M)<C^*(M-1)$ then record details of solution and goto RTS2.

RTS6. Ω=M-1.

The network is thus reduced from n nodes to q_1 nodes and the best single tandem is determined. The q_1 nodes are then reduced down to q_2 nodes and the best two tandems calculated. This process continues until the cost of M tandems exceeds the cost of M-1 tandems.

It is desirable, for reasons of accuracy, that as little reduction be done as possible. This hybrid algorithm will ensure that the largest value permitted value of q is used at all times. It also avoids any unnecessary repetition of reduction in the early stages.

6.2.4.2. Sources of Error

The fundamental problem in any approach which involves restricting the solution space is that the optimum may not lie in the subspace to which the search has been restricted. In the general case, the optimum is impossible to find under such conditions.

The situation with fixed topology tandem networks is not so bad. The solution space of all possible connection topologies is restricted (to the set of stars say). The purpose of the algorithm, however, is not to determine the connection topology, merely the best set of tandems. If the optimum connection topology is not contained in the set of stars, there is still a good chance that tandems produced within this restriction may be the same.

Errors, in this case, then occur if the set of tandems which produces the best star network is different from the set of tandems which produces the best overall network. This is still a problem, however. The refinement of a fixed topology solution is introduced in section 6.2.4.3 and is the subject of section 6.3.

6.2.4.1.2. Problems of Non-Convexity

The principle of testing values of M, only until $C^*(M)$ begins to increase depends upon the convexity (in a discrete sense) of the cost curve with respect to the number of transits. This is an excellent rule to apply since it holds for all but the most peculiar of situations (see chapter 7). In theory though it is not infallible. It is, in principle, possible for local minima to occur as a result of small blips in the C^* curve. This problem can be overcome as follows.

Instead of terminating the optimisation algorithm as soon as an M' is found such that $C^*(M')>C^*(M'-1)$, the decision is delayed until an M" is found such that $C^*(M")>PC^*(M^*)$ where P is a multiplicative factor ($P=1.05 \equiv 5\%$, $P=1.2 \equiv 20\%$, etc.) and M^* is the best value of M found so far. This implies that the cost must rise by a certain percentage before the algorithm stops. This helps to prevent blips on the curve making local minima appear global. Obviously, the larger the value of P the greater the security provided by this method (but more values of M are likely to be tested).

6.2.4.3. Fixed Topologies as Initial Solutions

Clearly there is a danger, its seriousness dependent upon the problem under consideration, that the optimum set of tandems, T_o, will not be found by restricting the solution space as suggested. The set of tandems found from the fixed topology method, T_f, may differ in one or more elements (tandems), $t_1, t_2, .., t_s$, $1 \le s \le M$ from T_o.

There is a good possibility that $T_o=T_f$ if the fixed topology has been chosen well. Alternatively, if $T_o<>T_f$, it is still likely that s, the number of tandems in error, is small. In the worst case, T_o and T_f may be very different indeed.

In general, some method is required for changing the (possibly) incorrect set T_f to the set T_o. The fixed topology method can thus be considered as the first step in a larger process in which an initial solution (T_f) is gradually modified and improved by an algorithm which makes step by step alterations to the network in such a way as to decrease the cost. If this algorithm is efficient, T_f will be transformed into T_o.

The next section discusses the principles and application of such a technique.

6.3. Initial Solutions and Perturbations

In a broad sense, the techniques employed to solve optimisation problems of all types can be categorised into a relatively small number of classes. There are simplification techniques in which a more convenient form of the problem is solved, reduction techniques such as the one proposed in section 6.1 and solution space restriction techniques of the form presented in section 6.2. One important class of methods are those which employ a form of step by step improvement approach to be known as <u>perturbation techniques</u>. In essence, these methods take an initial solution as input and apply small variations to its form. If any of these variations (perturbations) produce a solution which is more suitable than the original, in whatever sense the optimisation is being applied, then this new solution becomes the current solution. This process continues until no perturbations can be found, acting on the current solution, which give rise to an improved solution. The current solution then becomes the final solution.

This section presents a perturbation method of transit selection using the tandem set and connection network produced by the star tandem network restriction method as an initial solution. As with section 6.2, it will be assumed that a reduced q node network is being dealt with, although, of course, q may be equal to n and the reduced network may be identical to the original.

6.3.1. Basic Principles

Any optimisation problem requires, in precise terms, that a solution s^* be found from a valid solution space S such that $V(s^*)=\max_{s\in S}V(s)$, where V is the <u>value function</u> and V(s) is the value of the solution s in the sense in which the optimisation is taking place (for the NOP, the value of a network N is $V(N):=-C(N)$ where $C(N)$ is the cost of the network N).

Fig 6.p shows an idealised solution space. A perturbation method will apply step by step improvements to an initial solution. There are two distinct ways in which this initial solution can be produced.

1. Randomly selected

or 2. Chosen by some previous heuristic

Randomly selected solutions, clearly take no effort to produce but will, on average, be some distance from the optimum. This implies both that the perturbation method is likely to take longer to terminate and that there is a greater danger of the optimum not being found (see the discussion in section 6.3.3).

The extra work involved in producing a reasonable approximation by another heuristic is often justified in that the perturbation process is likely to be quicker and the initial solution has a better chance of being close to the optimum.

Each set of perturbations to the current solution s can be thought of as scanning a neighbourhood of s. Any improvement within this neighbourhood becomes the new current solution and the centre of the new perturbation neighbourhood. The perturbation method thus produces a series of these neighbourhoods, moving across the solution space S.

When a current solution is obtained such that the perturbation neighbourhood around it contains the optimum s^*, the process is nearly complete. s^* becomes the new current network and, by definition, no further perturbation neighbourhood will enclose any superior solution. The problem may be that a non-optimal solution s' may be encountered such that $V(s')<V(s)$ for all s within the perturbation neighbourhood. In this case, the process will proceed no further and s' will be given as the final solution instead of s^*.

The field of application of perturbation techniques is wide. Rothfarb et al. [1970] employ a spanning tree modification system to design off-shore gas pipeline systems and Lin & Kernighan [1973] use

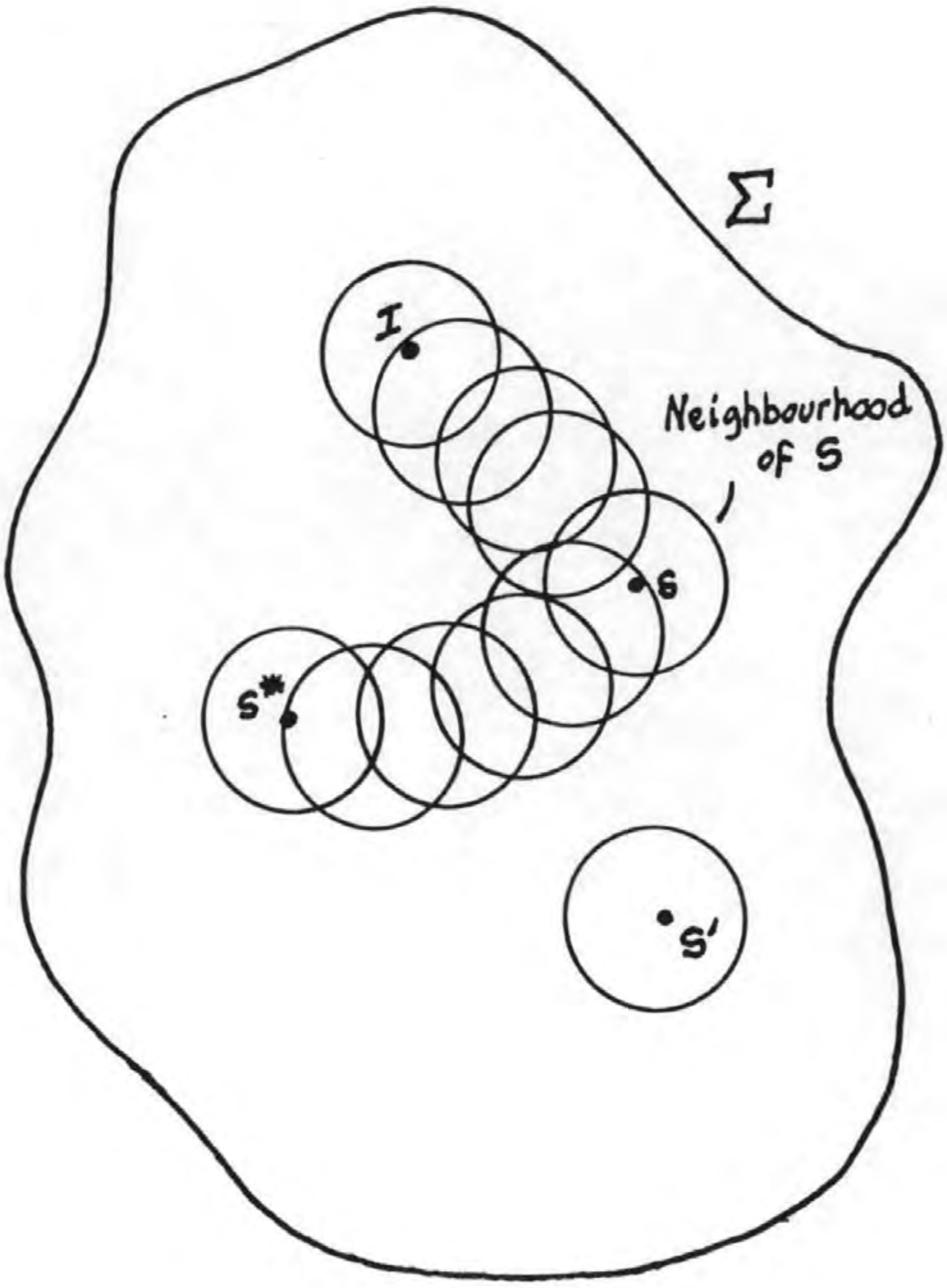


Figure 6.p. General Form of Solution Space

an edge swapping process to improve upon Travelling Salesman tours. Both are reported to give good results (although theoretical justification has apparently proved difficult). Numerical methods of systematically improving an algebraic solution are also well established (Gerald [1978]).

6.3.2. Applying Perturbations to an Initial Network Solution

The objective of the perturbation methods suggested here is still not to optimise the core network interconnection topology but to improve, if possible, upon the chosen set of tandems. Part of the perturbation procedure then must clearly involve changing the positions of the tandems (i.e. trying other nodes as tandems). This form of trial transformation, on its own, cannot, however, be expected to produce improvements since it is known that the chosen set of tandems is optimal for the restricted topology in question.

If the solution obtained by restricting the tandem network to a star is correct in the unconstrained case then clearly no perturbation method can expect to produce an improvement and will terminate after its first search. If the initial solution is not optimal, however, it will be in error precisely because the optimum does not involve a star tandem network. In addition to the movement of the tandem locations it is therefore necessary to vary the form of the connection topology as well. This, in perturbation terms, involves adding and removing sets of links from the tandem network.

The general approach will involve, at the primary level, moving each of the tandems, one or more at a time, around the other node locations within the network. For each selection of tandems, certain sets of links are added and removed as well. This will produce tandem network topologies which were not considered by the restrictive method. The algorithm will thus draw the tandem network away from the star form and may, in consequence, cause a different set of tandems to be preferred.

The following algorithm carries out the general perturbation method. In addition to the initial and final networks, two others are used. The current network is the best solution to date and the temporary network is the solution being examined during the various perturbation stages.

P1. Take the solution obtained by the tandem network restriction method as initial solution. Let the current network be the initial network.

P2. Let the temporary network be the current network. Select the first J set of tandems $J_1, J_2, .., J_x$.

P3. Select the first set of nodes $i_1, i_2, ..., i_x$.

P4. Move each tandem J_a, $a=1,2,..,x$, to node site i_a in the temporary network. Reconnect new tandems as for the current network. Connect each node to its nearest tandem.

P5. Select the first K set of tandems $K_1, K_2, .., K_y$.

P6. Connect each J_a to K_b, $a=1,2,..,x$, $b=1,2,..,y$ in the temporary network.

P7. Select the first L set of tandems $L_1, L_2, .., L_z$.

P8. Disconnect each J_a and L_b, $a=1,2,..,x$, $b=1,2,..,z$, in the temporary network.

P9. Calculate cost of temporary network. If the cost of the temporary network is less than the cost of the current network then take the current network to be the temporary network and goto P2.

P10. Reconnect the appropriate J_a and L_b, $a=1,2,..,x$, $b=1,2,..,z$, where necessary. If there are other L sets to be considered then reselect L set and goto P8.

P11. Disconnect the appropriate J_a and L_b, $a=1,2,..,x$, $b=1,2,..,y$, where necessary. If there are other K sets to be considered then reselect K set and goto P6.

P12. If there are other i sets to be considered then reselect i set and goto P4.

P13. If there are other J sets to be considered then reselect J set and goto P3.

P14. Take the final network to be the current network.

The J set, in each case, is the set of tandems to be moved and the i set, the set of node locations for them to be moved to. Links are added between all tandems in the J set and all tandems in the K set and removed between all tandems in the J set and all tandems in the L set. Adding a link between two tandems which are already connected has no effect nor does removing a link between two tandems which are already disconnected. The selection and reselection of the i, J, K and L sets can each be achieved using a minor variation of the algorithm in appendix A.

Clearly the values of x (the number of tandems moved at each stage), y (the number of tandems to which the moved tandems are connected) and z (the number of tandems from which the moved tandems are disconnected) will be of considerable importance in the operation of the algorithm. The larger the values of x, y and z, the more extensive the perturbations. This, in turn, decreases the possibility of the optimum being overlooked. This must, of course, be balanced against the extra complexity incurred by such improvements.

The simplest case (but still surprisingly effective) is with x=y=1 and z=0 (i.e. one transit is moved at each stage, one link at a time is added and no links are removed). Fig 6.q shows an example of the sequence for the case where x=y=z=1. The tandem J is moved to the node i, connected to K and disconnected from L. In this case, the perturbation process will continue until a complete tandem cycle, J,J+1,..,M-1,M,1,2,..,J-1, is tested with no improvement.

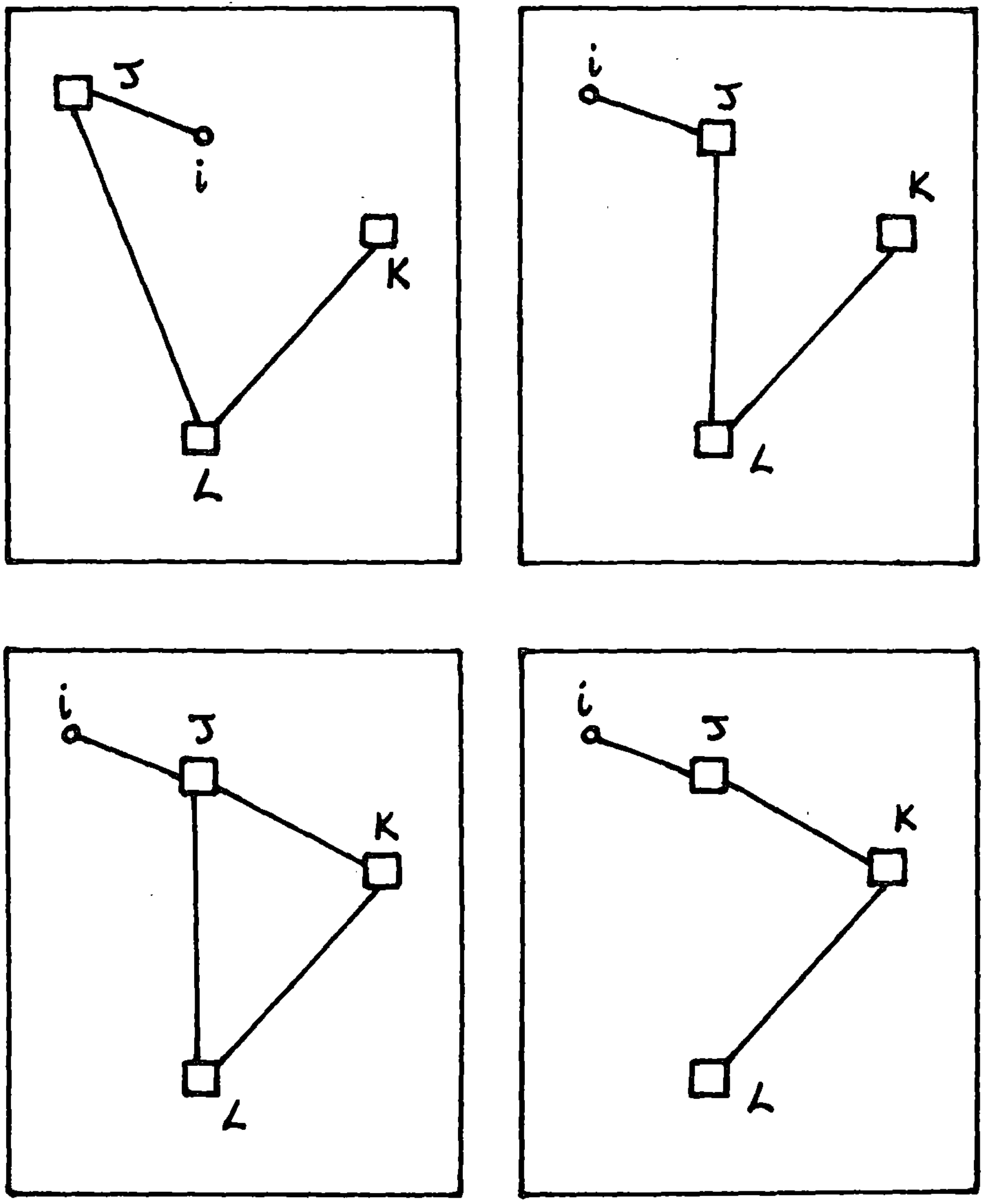


Figure 6.q. Tandem Perturbation Method: x=y=z=1

At each stage of this perturbation process, a check must be kept on the constraints governing the form of the solution. Constraints such as DOR and LOC must never be violated. Only valid networks can be adopted as new current networks. To ensure that the calculations are manageable, all traffic must be routed along the shortest path in all cases.

There is a simple modification that can be made to the above algorithm which greatly increases its power. This is to allow tandems to be added and deleted from the network in addition to simply being moved around.

The fixed topology algorithm will produce a set of M^* tandems where M^* is an approximation for Ω and the tandem set is an approximation for the optimum tandem set. The perturbation algorithm suggested in this section may improve the tandem selection but cannot, in this form, change the value of M^*.

Let the tandem set to be moved be $J_1, J_2, \ldots, J_{x'}$ and the set of node sites to which they are moved be $i_1, i_2, \ldots, i_{x''}$. If $x' = x''$ then there is no change to the existing algorithm and $x' > x''$ makes little sense. If however $x' < x''$, the x' tandems $J_1, J_2, \ldots, J_{x'}$ can be moved to sites $i_1, i_2, \ldots, i_{x'}$ as before and $x''-x'$ new tandems can be installed at sites $x'+1, x'+2, \ldots, x''$. These extra tandems can be connected into the existing network in a number of ways, one of the simplest being to use a single link to the closest existing tandem (unless the DOR or LOC are violated in which case a tandem can be systematically connected to its second nearest, third nearest, etc. until the constraints are satisfied). In this way the value of M can be increased.

Alternatively, the node site set $i_1, i_2, \ldots, i_{x''}$ may contain dummy markers i_∞. Any J_a moved to a site i_a where $i_a = i_\infty$ is, in practice, removed from the tandem network. All links incident to J_a are removed and any nodes connected to J_a are reconnected to the next nearest tandem. Traffic passing through the removed tandem should be rerouted via the shortest alternative path. In this way the value of M can decrease.

The procedures for generating node and tandem sets can easily be adapted to generate larger sets containing dummy sites. The above algorithm, together with these modifications, now provides an extremely general method of improving an initial solution.

When it is finally apparent that no more tandems may be added, removed or moved within the network in such a way as to produce an improvement then the process terminates. The number and location of the tandems can now be regarded as fixed. The connection topology and routeing strategy remain to be determined.

6.3.3. A Discussion of Perturbation Techniques

A solution to any NOP will be a network N. N will consist of a large number of parameters representing tandem locations, connection topology, etc. These parameters may be represented by the vector $\underline{n}$. The cost of the network N, C(N), can be thought of as a function $c(\underline{n})$ of the vector $\underline{n}$. If there are p elements in $\underline{n}$ then $c(\underline{n})$ forms a hypersurface in p+1-D hyperspace. Fig 6.r shows a 2-D representation of this with the $\underline{n}$ variables artificially represented as a single entity along the horizontal axis. The initial, final and optimum solutions to a perturbation process will correspond to points along this axis. The relationship between the initial and optimum solution can be seen to have a considerable effect on the chances of the final solution being the optimum. There are two basic cases to consider.

1. The initial solution is close to the optimum solution in the sense that either $dc/d\underline{n}<0$ for all networks $\underline{n}$ between $\underline{s}_1$ and $\underline{o}$ or $dc/d\underline{n}>0$ for all networks $\underline{n}$ between $\underline{s}_1$ and $\underline{o}$. Any system of perturbations should, under these circumstances, locate the optimum solution.

2. The initial solution is not close to the optimum solution in the sense that $dc/d\underline{a}=0$ for some $\underline{a}$ between $\underline{s}_2$ and $\underline{o}$. Convergence to the optimum will depend upon the level of perturbation being used. Inadequate perturbations may produce the locally optimum network $\underline{b}$ as the final network.

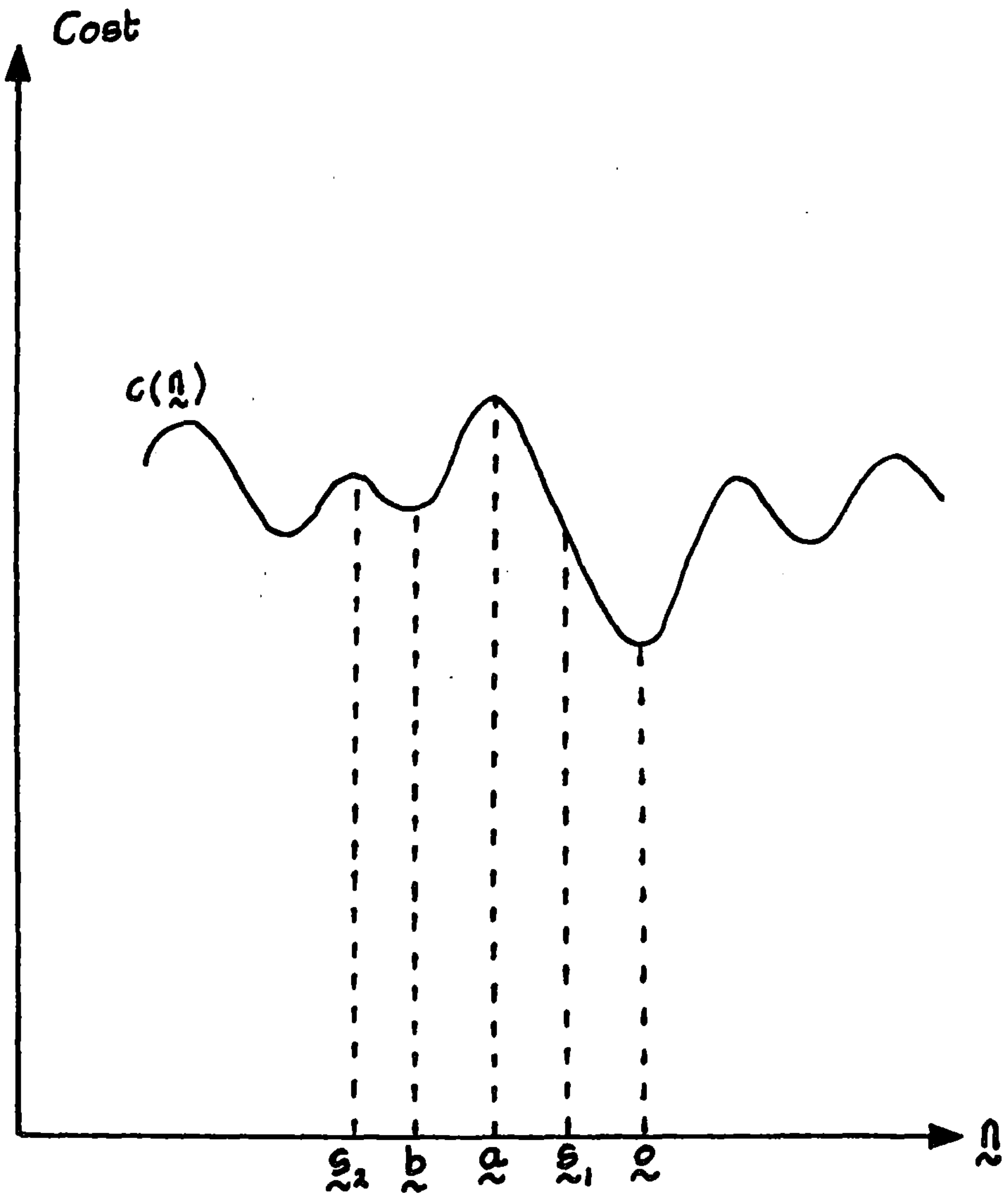


Figure 6.r. The c(n) Projection in Two Dimensions

Case one clearly presents no difficulties. It will be situations arising of type two that may give rise to errors. The perturbation method must be extensive enough to escape any local minima. Let the difference, $D(T_1,T_2)$, between two solutions T_1 and T_2 be represented by the pair (μ,σ) where μ is the number of tandems in T_1 but not in T_2, or vice versa, and σ is the number of links in T_1 but not in T_2, or vice versa. Then for an initial solution $\underline{s}_2$ to be transformed into a final solution $\underline{o}$, a sequence, $\underline{s}_2$, $\underline{t}_1$, $\underline{t}_2$, ..., $\underline{t}_p$, $\underline{o}$, must exist with $D(\underline{s}_2,\underline{t}_1)\leq(x,\min(y,z))$, $D(\underline{t}_j,\underline{t}_{j+1})\leq(x,\min(y,z))$ for all $j=1,2,..,p-1$ and $D(\underline{t}_p,\underline{o})\leq(x,\min(y,z))$ where $(a,b)\leq(c,d)$ is taken to mean $a\leq c$ and $b\leq d$. The existence of such a sequence, however, does not guarantee convergence in the general case.

Of course, even if the final vector $\underline{b}$ is not equal to the optimum vector $\underline{o}$, it may be that $\underline{b}$ and $\underline{o}$ differ only in those those parameters representing the connection topology. This is acceptable at this stage.

It would be possible, in theory, to make the perturbations so extensive that all possible values of $\underline{n}$ are covered. In practice, of course, this equates to an exhaustive search algorithm of the type it is hoped to avoid.

6.3.3.1. Multiple Starting Solutions

Many optimisation techniques which operate by transforming an initial solution into a final solution, in the way outlined in this section, can be improved by repetition with a number of different initial solutions. If the process is carried out u times with initial solutions $I_1,I_2,..,I_u$, it will produce v final solutions $F_1,F_2,..,F_v$ where $u\geq v$. The ratio u/v will increase as u increases since more duplication of final solutions will occur. If u is made large enough so that $u/v>2$ (say) then each final solution will, on average, have occurred twice and there is a good chance that the global optimum is among them (i.e. has occurred at least once). Selecting the best of the v final solutions will, in all likelihood, produce the optimum.

This approach can, if processing resources permit, be applied to the perturbation method suggested in this section. The different starting solutions will be tandem sets and connection topologies produced by different restriction methods at the fixed topology optimisation stage. Two obvious candidates exist for this purpose; the star and full mesh networks. A finite number of others can be produced in between these two extremes either at random or by insisting that each tandem is connected to at least two, three, .., M-2 others. This approach will, of course, increase processing time.

6.3.3.2. Complexity Analysis

It is, unfortunately, entirely impossible to predict accurately the running time of the perturbation process. It will depend, quite simply, upon the number of times that ways are found to improve the current network. Clearly, this depends, not only upon the initial solution and level of perturbation, but also upon the particular network in question. It is, however, possible to achieve some measure of the amount of effort expended at each stage.

Recall that x tandems are moved at each stage. These tandems can be chosen in ${}^{M}C_{x}$ ways and the node sites to which they are moved can be chosen in ${}^{q-M+x}C_{x}$ ways. There are ${}^{M-x}C_{y}$ ways of choosing the connecting tandems and ${}^{M-x-y}C_{z}$ ways of choosing the disconnecting tandems. The maximum number of networks considered is therefore given as ${}^{q-M+x}C_{x}\,{}^{M}C_{x}\,{}^{M-x}C_{y}\,{}^{M-x-y}C_{z}$. If the number of improvements is I (not known at the beginning of the process), the total number of networks examined will be bounded by $I\,{}^{q-M+x}C_{x}\,{}^{M}C_{x}\,{}^{M-x}C_{y}\,{}^{M-x-y}C_{z}$.

This, for larger values of x, y and z, is an extremely large number. It is not too important when M is small but, clearly, as M increases, x, y and z will, without option, have to be restricted.

It should, however, be noted that this maximum value only occurs when each improvement is found on the last perturbation at every stage - an extremely unlikely event.

6.4. Lookahead Techniques

A large number of optimisation methods operate by seeking courses of action which lead to large improvements in the value of the current solution. A suitably general term for this type of method is a greedy algorithm (defined in section 5.3.1.3.3 as one which seeks solely to maximise the improvement at each stage). Typically this may involve beginning with a starting solution and gradually changing it in such a way as to maximise the increase in value until such a time as no improvements are possible. Under certain circumstances, these techniques may be similar to the perturbation techniques of section 6.3 (see section 6.4.4). There is, in the general case however, an important distinction to be made between them.

1. Perturbation methods search for any improvement to the current solution. Any superior solution found in this way, immediately becomes the new current solution. The search process is consequently as simple as can be imagined but the changes made to the network as a result, may not be as accurate as might be wished.

2. Greedy methods consider every possible change which can be made to the current solution. The change which gives rise to the maximum improvement in value is put into effect. The search process is consequently more complex than for perturbation methods but there is a greater chance of a better decision being made at each stage.

Section 6.4.4 considers the relationship between perturbation and greedy methods. For the time being, however, they will be treated as separate. The purpose of this section is to develop a method of determining the tandem network connection topology as well as the traffic routeing strategy. A perturbation method was proposed in section 6.3 for the tandem set selection problem. In contrast, the problem in this section is approached using a greedy algorithm concept.

It should be noted, at this point, that if link costs were entirely linear with respect to distance and switch costs entirely linear with respect to port numbers, then the cheapest network would be the one in which all traffic flowed along direct links (i.e. a full mesh). In practice, however, this is not the case at all. In particular, there is a strong, and not unreasonable, tendency for the link cost per unit of traffic to decrease as the traffic increases. A result of this of this is that the first circuit of a link is often more expensive than subsequent links. These characteristics cause traffic rerouteing to be extremely desirable.

It will be assumed that the locations of the Ω tandems are established, either by a fixed topology method or by a fixed topology method followed by perturbation, and that there are $q-\Omega$ other nodes in the network where, as before, the complete set of q nodes may or may not represent a replacement network. Alternatively, the tandems may already actually exist in a real world situation. It will then be entirely unnecessary to calculate their optimum locations - only the optimum connection and routeing strategy will be relevant.

The approach adopted in this section is to attempt to extend the principles of greedy algorithms. Greedy algorithms differ from perturbation methods by doing more work at each stage to improve the accuracy at each stage (i.e. to decrease the chance of going astray). A method is now proposed which increases, further, the complexity of each stage to the eventual benefit of the final solution.

6.4.1. Principles of Lookahead Optimisation

Consider a typical greedy algorithm acting upon an optimisation problem beginning, at stage S_1, with an initial solution T_1. At stage S_1 there are c_1 changes, $C_{11}, C_{12}, \ldots, C_{1c1}$, which can be made to T_1 to transform it into $T_{11}, T_{12}, \ldots, T_{1c1}$ respectively. Suppose that change α produces the greatest improvement. Then the new current solution at stage S_2 is $T_2=T_{1\alpha}$. In the general case, at stage i, the maximum change, μ, will be made which transforms the solution T_i into solution $T_{i+1}=T_{i\mu}$.

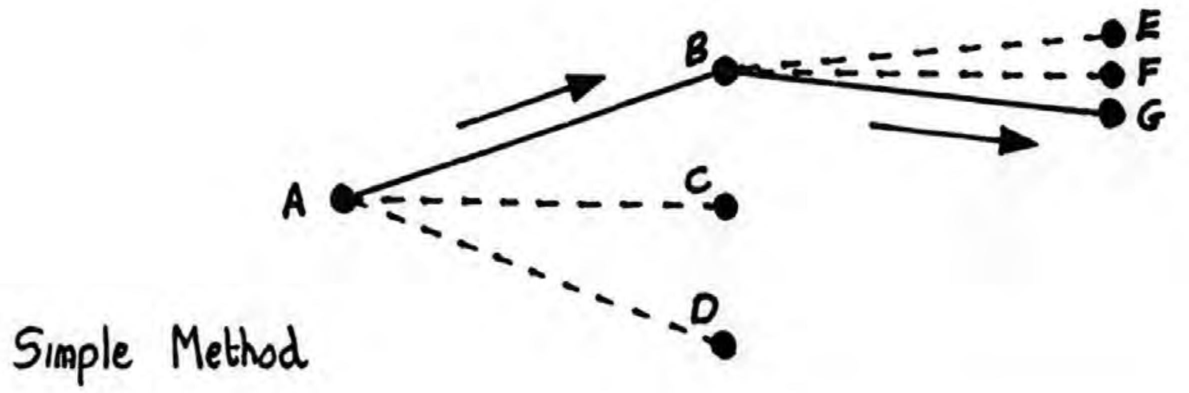
This step by step approach has one major drawback or, more precisely, one particularly serious source of error. The criterion for selecting change μ at stage i is based purely on maximising the value of $V(T_{i\mu})-V(T_i)$ $(\equiv V(T_{i+1})-V(T_i))$. The subsequent progression T_{i+2}, $T_{i+3}, \ldots, T_f$ (T_f being the final network after f stages), is entirely unknown. It is quite possible, and in some situations likely, that the optimum solution, T_o, is in a progression $T_{(i+1)'}$, $T_{(i+2)'}$, ..., T_o, where $V(T_{i+1}) \geq V(T_{(i+1)'})$ and yet $V(T_o) > V(T_f)$. At any stage the incorrect progression may be chosen by the greedy method. If this happens the mistake is, in the most general case, impossible to rectify. It would be far more acceptable if additional solutions within the progression T_{i+2}, T_{i+3}, ..., T_f were considered at stage i.

A <u>lookahead</u> method of optimisation <u>of degree r</u> is one in which the progression of solutions at stages S_{i+1}, S_{i+2}, S_{i+3}, ..., S_{i+r} is considered at stage S_i. The solution $T_{(i+r)'}$ is chosen such that $V(T_{(i+r)'}) \geq V(T_{i+r})$ for all other solutions T_{i+r}. The solution chosen for stage S_{i+1} is the solution which, at stage S_{i+1}, is in the progression eventually containing $T_{(i+r)'}$. The process thus still steps through the progression of solutions one at a time, but it does so with far more information about future developments than the simple greedy method.

Fig 6.s shows an example of this effect diagramatically. The simple greedy method (lookahead with r=1), at position A, considers only the merits of B, C and D whereupon it moves to position B and in turn to position G having only considered E, F and G. The lookahead method of degree 2, however considers the values of E, F, G, H, I, J, K, L and M and, finding L to be of greatest value, moves to D before repeating the process from this new position.

A chess machine works on this principle. The value of all possible board positions up to a number of stages ahead are considered but only one move can be made at a time (there is a slight difference - a lookahead optimisation algorithm seeks the overall maximum value whereas a chess machine seeks to minimise the maximum loss since it has to deal with unknown input between each stage). Other practical applications of lookahead techniques are, to date, limited. Lin & Kernighan [1973] propose, very briefly, a lookahead extension to their link swapping algorithm for the TSP.

The principle of lookahead techniques is that they can overcome the problems of only considering the next stage (lookahead with degree 1). The greater the degree of lookahead, r, the less myopic the method



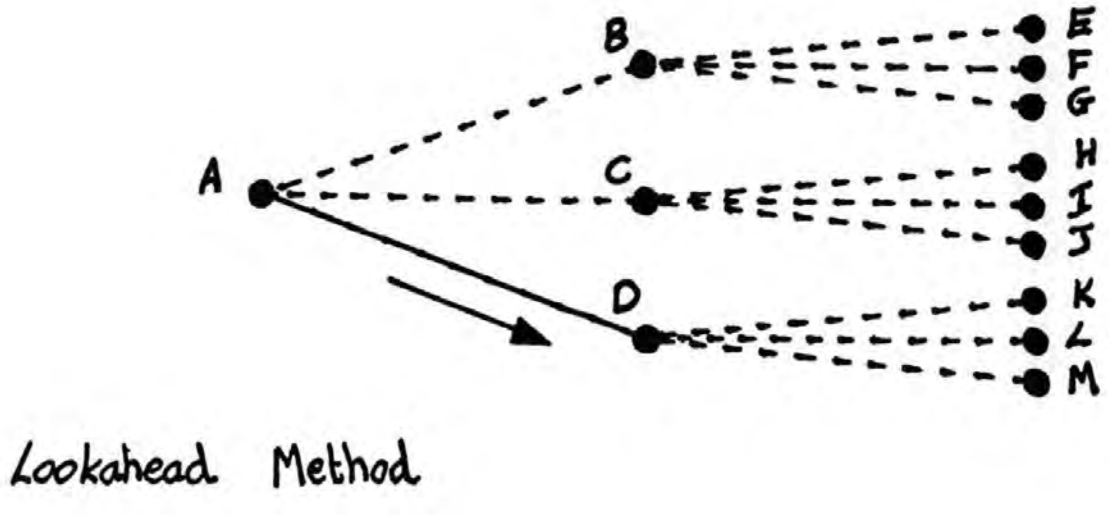


Figure 6.s. Simple Method and Lookahead Method

becomes and the smaller the chance of error. This extra security must be balanced against the extra complexity incurred. A similar situation exists as with the perturbation method. It is possible, in principle, to make r large enough so that the progression of solutions examined was complete (i.e. $1+r=f$) but, in practice, this is only an exhaustive search method in thin disguise.

6.4.2. Tandem Network Optimisation

Part of the double drop method of section 5.5.2 was a process by which links were systematically removed from the tandem network in conjunction with the tandems. Consider a simple variation of this approach in which the tandems remain fixed but the links do not. The following algorithm will produce an approximation to the optimum connection topology for $\Omega>2$ (if $\Omega\leq2$ the network is trivial).

D1. Begin with each of the Ω tandems connected to all of the other $\Omega-1$. Let all traffic between any pair of tandems flow along the direct link between them. Connect each of the $q-\Omega$ remaining nodes to their nearest tandem.

D2. maximum=0. Choose a tandem I.

D3. Choose a tandem J.

D4. If there is no link between I and J then goto D10.

D5. Choose a tandem K.

D6. If there is no link between I and K or if there is no link between J and K then goto D9.

D7. Calculate the saving of removing the link (I,J) and rerouteing the traffic via K. If saving≤maximum then goto D9.

D8. Set I':=I, J':=J, K':=K and maximum=saving.

D9. If there are any more tandems K to test then reselect K and goto D6.

D10. If there are any more tandems J to test then reselect J and goto D4.

D11. If there are any more tandems I to test then reselect I and goto D3.

D12. If maximum>0 then remove link (I',J'), reroute the traffic via node K' and goto D2. Otherwise stop.

This algorithm starts with a full mesh connection and systematically drops links according to the maximum improvement in cost. It is an extremely powerful method of tandem network design since it also determines the routeing strategy at the same time (this being the most complex of all the component problems of the NOP). Moreover, since the topology of the tandem network and the routeing strategy are so interdependent, it makes a lot of sense to attempt to deal with them within the same routine. As always, it is necessary to ensure, at each stage, that no link is removed which causes any of the solution constraints, such as DOR or LOC, to be violated. This simple, yet powerful, method forms the basis for the more general technique presented in this section.

A familiar problem exists with this approach. A link may be removed at an early stage, through the consideration of insufficient information, which could prove to be of considerable importance later. The method is clearly greedy, however, and therefore the lookahead extension suggested in section 6.4.1 may be applied in an attempt to minimise the chance of removing an incorrect link.

The lookahead process will involve considering a series of link removals up to r stages into the future. To do this, an array $\underline{L}$ must be defined where

$$\underline{L}^T = (I_1,J_1,K_1,I_2,J_2,K_2,\ldots,I_s,J_s,K_s,\ldots,I_r,J_r,K_r)$$

It is then intended that, at s stages into the future, the link between I_s and J_s will be removed and the traffic rerouted via K_s. As with perturbation, the implementation of such a technique requires that, in addition to the initial and final networks, a current and temporary (test) network are required. The following algorithm will then carry out a lookahead link removal process of degree r with the respect to the tandem network.

L1. Let the initial network be such that each of the Ω tandems is connected to all of the other Ω-1. Let all traffic between any pair of tandems flow along the direct link between them. Connect each of the q-Ω remaining nodes to their nearest tandem.

L2. Let the current solution be the initial solution.

L3. maximum=0. Select the first $\underline{L}$ vector. r':=0.

L4. Let the temporary network be the current network.

L5. r':=r'+1. If there are links between $I_{r'}$ and $J_{r'}$, $I_{r'}$ and $K_{r'}$ and $J_{r'}$ and $K_{r'}$ in the temporary network then remove the link $(I_{r'},J_{r'})$ and reroute the traffic via $K_{r'}$.

L6. Calculate the saving in cost of the temporary network over the current network. If saving>maximum then set I':=I_1, J':=J_1, K':=K_1 and maximum=saving. If r'<r then goto L5.

L7. If there are any more $\underline{L}$ vectors to test then generate next $\underline{L}$ vector and goto L4.

L8. If maximum>0 then remove link (I',J') from the current network, reroute the traffic via K' and goto L3. Otherwise let the final network be the current network and stop.

This algorithm is a slight extension of the formal lookahead method of degree r in that it considers all networks r' stages into the future where $1 \le r' \le r$. Some system of initialising and updating the $\underline{L}$ vector is required within this algorithm. A procedure for such a purpose is presented in appendix A.

The chance of removing an erroneous link at an early stage will be decreased by using this technique for $r>1$. As with the simple version, the routeing strategy is determined at the same time as the connection topology. In this way two potentially enormous problems have been combined and solved by the one process. As always, links may only be removed, or traffic rerouted, if such an action does not cause any of the solution constraints to be violated.

6.4.2.1. Routeing Strategies for Predetermined Topologies

This section is somewhat removed from the main study of section 6.4. It is included to deal briefly with a simple restriction to the tandem network optimisation problem.

The situation will sometimes arise in which a network already exists in which all topological decisions have been made. The only work to be carried out is to determine the traffic routeing strategy for this network. This will happen typically when it is required that the operation of a large and unsatisfactorily performing network be reconsidered. The required routeing strategy may be determined, very simply, as follows.

TR1. Any pair of tandems having a direct connection will have the traffic between then routed along that link.

TR2. For all pairs of tandems which do not have a direct connection, do the following:

TR2.1. Suppose that the traffic is routed along an imaginary link between these two tandems.

TR2.2. Determine the shortest path (in terms of number of links) between these two tandems. Let the length of this path be r.

TR2.3. If the path of length r is unique then reroute the traffic accordingly. Otherwise choose the one of least cost before rerouteing the traffic.

If the shortest path between the two unconnected tandems is unique then there is no choice to be made. If, however, there is more than one valid path of length r then the one which provides the rerouteing of least cost is chosen. This method is clearly heuristic since no account is taken of the effect of various traffic flows upon each other. It does, however, overcome a potentially awkward problem (the enormous number of possible routeing strategies for a given topology).

It is not advantageous, in most cases to use lookahead methods of rerouteing (simple comparison being easier). However, if there are an extremely large number of paths of length r, lookahead methods of degree r may be preferred in which the traffic is rerouted, in stages, from the imaginary link to two others, three others, etc..

6.4.3. A Discussion of Lookahead Techniques

Lookahead methods provide an appreciable improvement over simple greedy algorithms. More information is considered at each stage with the result that better long-term solutions are likely. For any degree of lookahead (r), however, there is still a limit to the amount of foresight that can be provided. r cannot be made too large or the process will take too long to complete and it is still possible, albeit less likely, that some information more than r stages into the future is relevant to the choice taken at the current stage. This section discusses the practical problems in the choice of r and the resultant possibility of error.

6.4.3.1. Complexity of the Method

The number of possible actions (link removals and traffic rerouteings) will decrease progressively as links are removed. If at any stage, there are L links remaining in the network, there will be a maximum of $L(\Omega-2)$ options for the next change (the traffic on each link can be rerouted via up to $\Omega-2$ other tandems - not all such rerouteings will be possible but, in the general case, there is no way of knowing). The maximum value of L, at the start of the process, is $\Omega(\Omega-1)/2$ and the number of changes is exactly $\Omega(\Omega-1)(\Omega-2)/2$. This, being cubic in Ω is quite acceptable. Once a link has been removed, however, it can clearly not be removed again (taking away $\Omega-2$ choices at the next stage) and it can have no traffic rerouted onto it (removing $2(\Omega-2)$ choices from consideration). The number of choices at the next stage is therefore $\Omega(\Omega-1)(\Omega-2)/2-3(\Omega-2) = (\Omega-2)[\Omega(\Omega-1)-6]/2$ which is still cubic in Ω. This number of choices will be available following each of the $\Omega(\Omega-1)(\Omega-2)/2$ options at the first stage. Extending this argument over successive stages shows that the number of choices at r stages into the future is still potentially of the order $r^{\Omega x\Omega x\Omega}$. This is only acceptable for small values of r or small values of Ω but not large values of both. As Ω increases, r is restricted severely. It important to ensure that such a constraint does not affect the accuracy of the solution adversely. The results in chapter 7 provide practical justification of this. The following section approaches the question from a theoretical point of view.

6.4.3.2. Possible Errors

Consider the worst case of all, $r=1$. The lookahead method is now no more than a simple greedy algorithm. In general, the initial solution network N_1, at stage S_1, is transformed into solution N_2, at stage S_2, by choosing the maximum improvement after r stages (and $r=1$), and subsequently into N_3, N_4, ..., N_f. There may be, however, a solution $N_{f'}$, such that $C(N_{f'})<C(N_f)$, where $N_{f'}$ is part of a sequence $N_{2'}$, $N_{3'}$, ..., $N_{f'}$. There must be some k, $2\leq k\leq f$, for which $C(N_{1'})>C(N_1)$

$$\begin{tikzcd}
\text{Stage} & \text{Chosen Progression} & \text{Optimum Progression} \\
S_1 & N_1 \arrow[d, dashed, "\alpha_1"] & N_1 \arrow[d, dashed, "\alpha_{1'}"] \\
S_2 & N_2 \arrow[u, bend left] \arrow[r] \arrow[d, dashed, "\alpha_2"] & N_{2'} \arrow[u, bend left] \arrow[d, dashed, "\alpha_{2'}"] \\
S_3 & N_3 \arrow[u, bend left] \arrow[r] \arrow[d, dashed, "\alpha_3"] & N_{3'} \arrow[u, bend left] \arrow[d, dashed, "\alpha_{3'}"] \\
 & \vdots \arrow[d, dashed, "\alpha_{k-2}"] & \vdots \arrow[d, dashed, "\alpha_{k-2'}"] \\
S_{k-1} & N_{k-1} \arrow[r] \arrow[d, dashed, "\alpha_{k-1}"] & N_{k-1'} \arrow[d, dashed, "\alpha_{k-1'}"] \\
S_k & N_k \arrow[u, bend left] \arrow[d, dashed, "\alpha_k"] & N_k \arrow[u, bend left] \arrow[l] \arrow[d, dashed, "\alpha_{k'}"] \\
 & \vdots \arrow[d, dashed, "\alpha_{f-1}"] & \vdots \arrow[d, dashed, "\alpha_{f-1'}"] \\
S_f & N_f & N_{f'} \arrow[l]
\end{tikzcd}$$

$\longrightarrow \equiv$ 'is a better solution than'

Figure 6.t. Progressions of Network Solutions in Lookahead Method

for $i<k$ and $C(N_{i'})<C(N_i)$ for $i\geq k$. Clearly, for this to happen, $k>r$. Fig 6.t shows this effect diagrammaticlly.

6.4.4. A Comparison between Lookahead and Perturbation Techniques

The fundamental distinction between perturbation methods of optimisation and lookahead techniques is that, at each stage, the first improvement to a solution is adopted in the first case whereas the greatest improvement over a series of tests is taken in the second. It is perfectly possible to change the specifications of either method.

1. A perturbation method may carry out a series of perturbations at each stage, recording the current greatest improvement at each turn. At the end of each series of tests, the best change can be made permanent and the process repeated for the next stage.

2. A lookahead method may consider networks up to r stages into the future but act upon the first improvement encountered instead of the best.

The second possibility is unworthy of too much consideration since it totally negates the entire purpose of looking ahead more than one stage. The first, however, allows an interesting comparison to be made.

If the perturbation method searches for the best option at each stage then there is clearly a distinct similarity between perturbation and lookahead techniques.

1. Both transform an initial solution into a final solution via a number of current solutions.

2. Both perform a series of tests on possible changes to the current solution in order to determine the next current solution.

A distinction remains, however. A perturbation method will transform a current solution directly into the new solution which it has found to give greatest improvement. A lookahead process, on the other hand, will only make a single change. It would, of course, be possible to restrict the perturbation method to making one change at a time but, in practice, it is difficult to decide which single change this should be. Such a decision is possible for the lookahead method due to the sequential way in which prospective solutions are generated.

This difference is really caused by the situation in which both techniques are used. With fixed tandems it is simple for the lookahead method to drop links in a sequential manner. The lookahead method of link removal may thus be regarded as a form of perturbation in which the tandems are fixed. The complete process then becomes a two stage perturbation process. Firstly, perturbation is carried out on tandem locations and links alike. At a certain point, the tandems become fixed and a link perturbation (lookahead) process continues.

There are two ways, other than the fixing of the tandems, in which the second process differs from the first.

1. The connection topology at the end of the tandem perturbation process is replaced by a full mesh before link perturbation begins.

2. Links can only be removed at each stage, and not added.

Section 6.5.2. suggests some enhancements to the compound algorithm presented in the next section. One of these (section 6.5.2.3) involves dealing with these two anomalies on the way to producing an improved optimisation algorithm for tandem network connection topology and traffic routeing.

6.5. A Composite Approach

The preceding sections of this chapter have presented a number of methods for dealing with various component problems of the general

NOP. Each can, if required, work independently of the others in certain situations. Usually a number of techniques have been suggested for each section, each being a variation on the same theme. The purpose of this final section is threefold.

1. To provide a summary of the major methods proposed in a concise form.
2. To construct a single compound algorithm, comprising a number of stages for the complete optimisation of a network.
3. To propose certain extensions to the algorithm for use in special cases.

The first two objectives are dealt with in section 6.5.1 and the third in section 6.5.2. In general, the compound algorithm will employ a single technique from each of the previous sections. This approach to network optimisation is also discussed in Grout, Sanders & Stockel [1987a & 1988].

6.5.1. Practical Network Optimisation

Chapter 7 contains results of a large number of tests performed upon network problems, both real and fabricated. Some of these results are concerned with individual component problems but many are obtained using the complete optimisation algorithm outlined in this section. The entire process from input data to final solution is given, mainly in terms of the techniques discussed in the preceding sections.

6.5.1.1. Input Data

The input data to a general NOP will be as described in section 5.1.1.1. n nodes are described by (x,y) coordinates in the plane and a number of traffic matrices describe the traffic flow between them. Other information such as the costs of various pieces of available equipment, performance constraints (GOS) and structural constraints

(DOR, LOC, etc.) may also be required. In the general case, n may be very large (up to a thousand nodes).

6.5.1.2. Reduction

It is not practical to deal with a large number of nodes in a direct manner. The amount of data alone would exhaust the storage resources of some machines and the processing time would be considerable.

The first step then is to replace the n node network by a q node representative one (in principle but see section 6.5.1.3). This is achieved using the step by step weighted reduction process of section 6.1.2.2. This produces q replacement nodes at various centres of heavy traffic. In addition, the traffic matrices must be reduced, in the manner of section 6.1.2.4, so that the traffic characteristics of each node match the combined characteristics of the nodes it has replaced. The location of each replacement node will, in general, be a a greenfield site (section 6.1.2.5). Each is moved to its appropriate nearest real site. It will be from these replacement nodes that the tandems are selected.

6.5.1.3. Tandem Selection

The tandem selection process is comprised of two sections. Firstly, an initial set of tandems is determined through constraining the connection topology to take the form of a star (section 6.2) and secondly, a series of perturbations are applied to this solution in a quest for improvements (section 6.3).

In practice the fixed topology algorithm operates alternately with the reduction process. The n nodes are reduced sufficiently for the optimum single tandem to be approximated. The network is then reduced as far as is necessary to deal with two tandems, and so on. When the cost begins to rise, the optimum is taken to have been passed. The tandem set produced in this way is passed to the perturbation process for refinement.

The perturbation of tandems and links allows the constraints of star topology to be lifted. A (possibly) new set of tandems is produced. The number and locations of these Ω tandems are now taken to be fixed. It remains to determine the connection topology of the network.

6.5.1.4. Tandem Network Optimisation

This process begins with Ω tandems together with their traffic matrices. Initially the tandems are all connected directly together. Links are then systematically dropped using the lookahead technique of section 6.4. At the same time the routeing strategy of the final network is determined since the removal of each link requires the redistribution of traffic carried by that link.

Once this process is completed, the tandem locations are established along with the connection topology linking then together and the routeing strategy. The complete process is finished by calculating the best way to connect each node to its nearest tandem.

6.5.1.5. Local Network Optimisation

This section has not been discussed before since its solution is trivial. It is required that the type and size of each link from node to tandems is found. This simply involves using Erlang B tables to determine the size of each link and calculating the cost of each option in order to choose the cheapest. The rest of the network is not affected.

6.5.1.6. Flexibility of the Compound Algorithm

Fig 6.u shows the complete algorithm in block form. There are, in general, a number of versions of each component which may be used. Even once a fixed method is decided upon, however, such as the one presented in sections 6.5.1.1-5, there are still three distinct areas of flexibility in the operation of the algorithm.

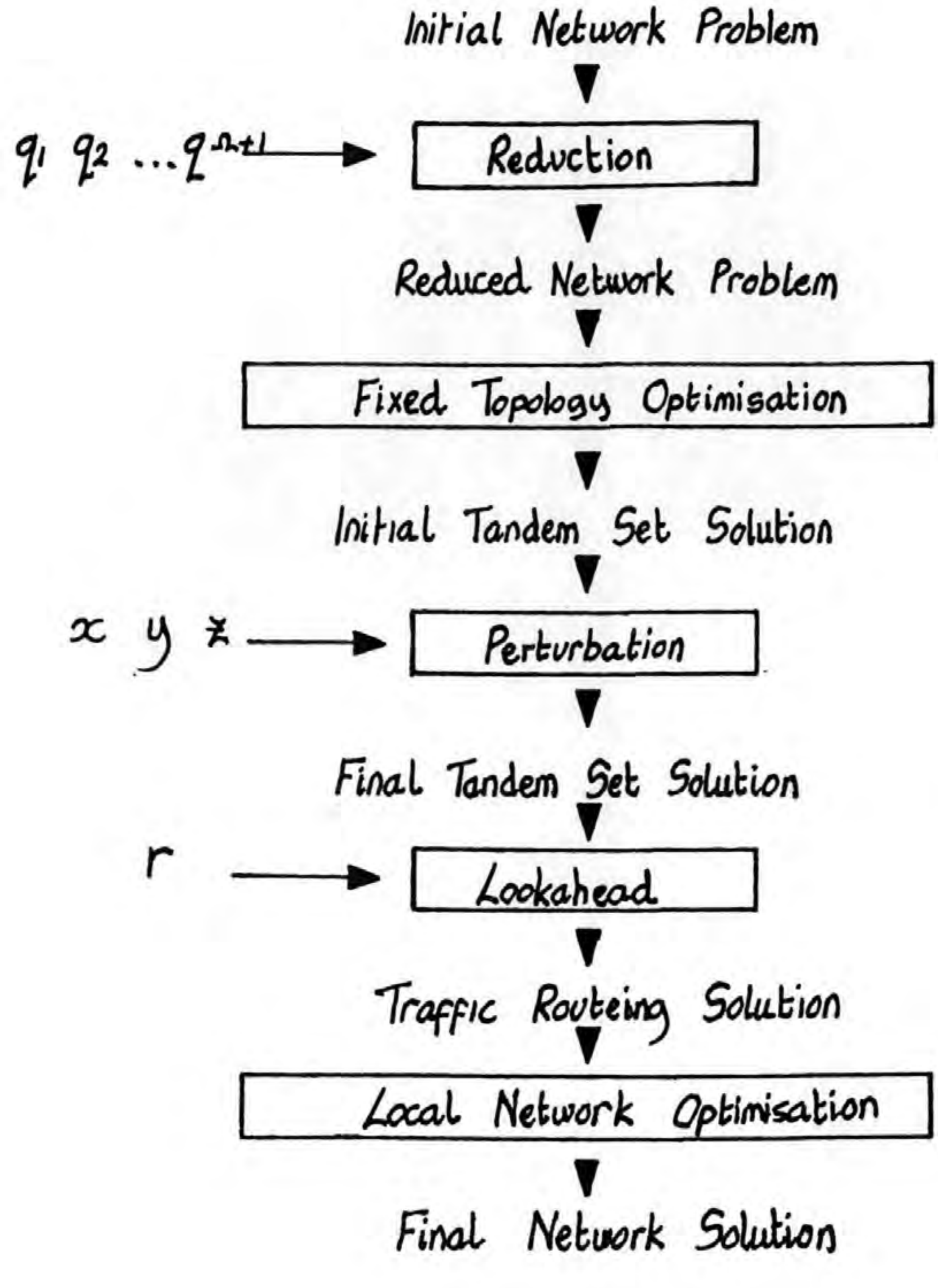

Initial Network Problem
$q_1\ q_2 \ldots q^{n+1}$
Reduction
Reduced Network Problem
Fixed Topology Optimisation
Initial Tandem Set Solution
$x\ y\ z$
Perturbation
Final Tandem Set Solution
r
Lookahead
Traffic Routeing Solution
Local Network Optimisation
Final Network Solution

Figure 6.u. Compound Optimisation Algorithm

1. The values of q_1, q_2, ..., $q_{\Omega+1}$, the number of nodes in the replacement network after the reduction stage for each value of M to be tested. Clearly the larger the value of q, the less chance there will be of errors in the latter stages of reduction. However, this will, of course, increase the amount of effort which has to be expended in the tandem selection stages.

2. The values of x, y and z, the number of tandems moved, connected and disconnected at each stage of the reduction process. The larger the values of x, y and z the greater the scope of the perturbation and the smaller the chance of missing a superior solution. Again, however, this will clearly result in an increase in processing time.

3. The value of r, the degree of lookahead in the tandem network connection topology optimisation section. As with the previous two, a large value of r will provide the algorithm with considerably greater foresight and give a better chance of finding the best progression of solution networks towards the optimum.

The values of q_1, q_2, .. ,$q_{\Omega+1}$, x, y, z and r chosen, in any given case, will depend primarily upon three things.

1. The network problem in question and, in particular, its size and traffic flow. Large, heavily loaded networks are likely to require more tandems than small lightly loaded ones. These, in turn, will be more difficult to optimise.

2. The processing resources available for the implementation of the algorithm. Large powerful machines will be able to deal with greater values of q_1, q_2, .., $q_{\Omega+1}$, x, y, z and r than smaller ones.

3. The desired trade-off between the accuracy required of the

solution and the amount of time and processing resources it is likely to take. Obviously, more accurate solutions will take longer to achieve.

Practical values of q_1, q_2, ..., $q_{\Omega+1}$, x, y, z and r based on actual observations are contained in chapter 7.

6.5.1.7. Complexity of the Compound Algorithm

It is difficult, in the general case, to provide a measure of the complexity of the complete process. This is because the number of successful perturbation and link removal stages is entirely unknown. Furthermore, the value of Ω is flexible so the amount of work which has to be done at the fixed topology stage is variable.

It is possible, however, to obtain a general upper bound for the complexity. The first stage of the compound algorithm will be to reduce the n initial nodes to q_1 nodes. The complexity of this section is $O(n^2 \log n)$. From this stage onwards, irrespective of future developments in the optimisation, there must be a maximum number of steps, C, taken in the extreme case since q_1 is always the same in all cases. The upper bound on the complexity of the algorithm is therefore $O(n^2 \log n+C) = O(n^2 \log n)$. The value of C, in practice however, should not be discounted lightly and if the number of steps is given by $Kn^2 \log n$ then it is possible for K to be extremely large indeed. In general the values of q_1, q_2, ..., q_{M+1} should be chosen with careful reference to the processing resources available and the table of section 6.2.4.1.

6.5.2. Extensions to the Compound Algorithm

This final section of chapter 6 investigates some improvements which can be made to the compound algorithm presented above under certain circumstances. The technique proposed in section 6.5.2.3 is, in fact, a candidate for use under any circumstances if a second method of tandem network topology calculation is required.

6.5.2.1. Windowing Techniques

The complexity of the reduction algorithm, $O(n^3)$ or $O(n^2 \log n)$, is indeed polynomial and therefore acceptable in most practical circumstances. For extremely large numbers of nodes (several thousand maybe), however, even the reduction process, as it now stands, may take too long to run.

Suppose that the maximum number of nodes that can be practically dealt with by the reduction process is n_m and that a particularly large network problem involves $s>n_m$ nodes. Possibly the only realistic way of dealing with such a problem is through a technique known as <u>windowing</u>. The region of the plane, R, containing the s nodes is divided arbitrarily into uniform sub-regions R_i, $i=1,2,\dots,t$, where t is large enough to ensure that $s_i<n_m$, $i=1,2,\dots,t$ where s_i is the number of nodes in region R_i.

Reduction then takes place individually within each sub-region in turn. The s_i nodes in each sub-region are reduced down to n_m/t independently of the other sub-regions. The divisions are then discarded and the reduction process resumes, as normal, on the entire network (which now contains n_m nodes).

The artificial boundaries at the early stages are likely to have little effect since the distances between replaced nodes will be extremely small until some considerable time after these boundaries have been removed.

6.5.2.2. Concentrators Within the Local Network

The network solution produced by the compound algorithm given in this section is basically a two level structure (tandems at the top level and nodes at the lower level). This will be satisfactory in most cases but, if required, a third level can be incorporated by the use of concentrators beneath the nodes within the local networks (the tandem groups) as shown in Fig 6.v.

The techniques needed to provide such a structure have already been discussed - namely the variable cost extensions to the add and drop methods (Bahl & Tang [1972]) proposed at the end of section

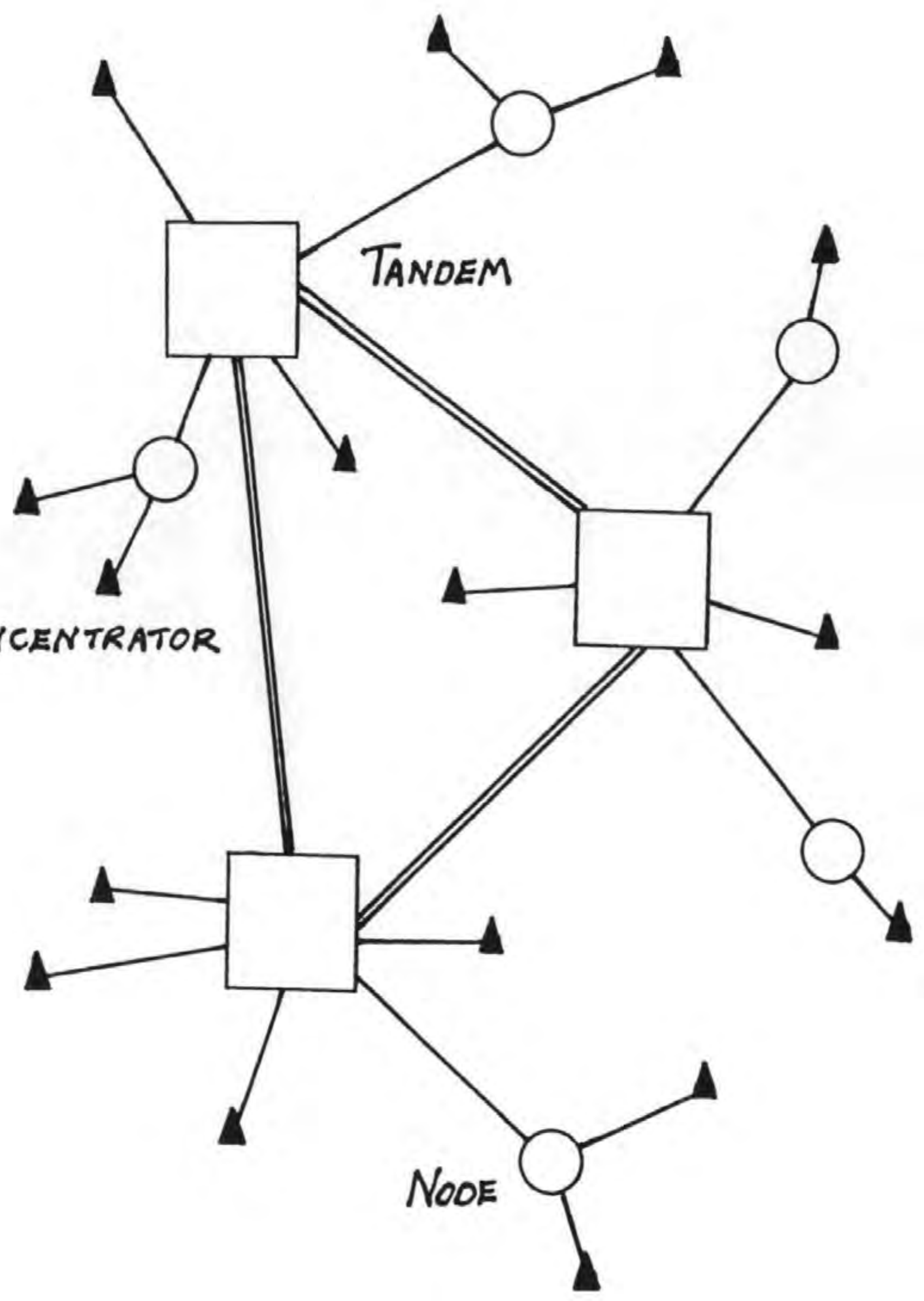


Figure 6.v. A Three Level Network

5.4.1.1 and implemented in section 5.4.1.4. One or other of these techniques can be applied, in turn, to each of the Ω tandem groups. For this purpose, the tandem can be considered as the central node and (in the case of the drop algorithm) nodes can be systematically dropped from the local network to become concentrators (a variation of the method suggested in section 5.4.1.4). The required traffic flow from each node to tandem (or from each concentrator to node or tandem) can be found from the traffic matrices.

6.5.2.3. An Improved Tandem Network Optimisation Algorithm

The existing process for determining the tandem locations and connection topology consists of a tandem and link perturbation section followed by a sequence of link removals starting from a fully connected tandem network. The following algorithm is a similar two stage method which differs from the original by preserving the smoothness of transition from the first section to the second.

FIRST SECTION

P1. Begin with the network structure produced by the fixed topology method.

P2. Implement the tandem and link perturbation process in order to produce a new network structure.

P3. Fix the number and locations of the tandems.

SECOND SECTION

L1. Begin with the network structure produced by the tandem and link perturbation method.

L2. Implement a lookahead method of link removals <u>and additions</u> to perturbate the connection topology and produce a new network structure.

L3. Fix the links within this network.

The first part of the process is the same as before. The second is different in that the lookahead process starts from an intermediate structure instead of a full mesh. From this position, links can be added as well as removed. Clearly the degree of lookahead must exceed the previous level of perturbation for any further improvements to become evident. This, however, is possible as a result of the work saved in applying perturbations to the tandem locations.

To add a link to the network involves the reinstatement of that link together with the traffic between the appropriate tandems rerouted along that link.

6.5.2.4. Parallel Processing Applications

This chapter concludes with a few brief comments about the possibility of extending the techniques presented here to situations in which parallel processing (PP) capabilities are in evidence. It is likely that such machines will be found in widespread use in the near future (Sumner [1986]) and it is felt that it is reasonable to consider the scope for implementation of each section of the compound algorithm.

1. The reduction process is not really suitable for PP techniques (unless the windowing techniques mentioned in section 6.5.2.1 are employed). The common method is entirely sequential in that each stage of reduction has to be completed before the distances can be compared for the next.

2. The fixed topology optimisation section, on the other hand, is ideal for PP implementation. The process is purely intended to test a large number of independent solutions. As many of these can be processed at once as required. This will produce a rare instance of perfect PP.

3. PP for the tandem network perturbation section can be

considered in two different ways depending on whether the first or greatest improvement to the current solution is adopted. If the first improvement is adopted then the value of PP is limited since it will be likely that some improvement will be found early at each stage (except at the latter stages when there are few improvements to be found). If, however, the best improvement is taken, then PP can significantly speed up the search process since each perturbation can be made independently of the others. Perfect PP again results.

4. Similarly, the various sequences of link removals and traffic rerouteings are independent of one another. PP can allow a number to be tested simultaneously, again producing perfect PP operation.

It is interesting to note that the only section of the compound algorithm which cannot be significantly improved by PP is the reduction process. This is extremely convenient since it is this section it would be most desirable to restrict. If PP resources were to become powerful enough to permit it, the reduction process could be shortened (i.e. the value of q made larger) and more work carried out at the subsequent stages, making use of their particular suitability for PP.

CHAPTER 7

RESULTS

'Write that down,' the King said to the Jury, and the Jury eagerly wrote down all three dates on their slates, and then added them up, and reduced the answer to shillings and pence

Lewis Carroll

Alice's Adventures in Wonderland, 1865

Chapters 5 and 6 presented a number of methods, both established and original, for solving various sections of the NOP. Wherever possible, some discussion was entered into with respect to the accuracy of each technique but, in some cases, probabilistic, or even worst case, analysis appeared impossible (as has often been the case with heuristic methods of solution - Lin & Kernighan [1973] for example). In these circumstances, empirical testing provides the only practical way of determining the value of any such approach. Even in those cases where rigid analysis is possible, results of practical experiments are usually appreciated for verification. This chapter contains the results of some appropriate numerical tests performed on various techniques for the NOP.

7.1. Results of Individual Experiments

This section records details of those experiments performed on various algorithms for the solution of certain components or restrictions of the general NOP. They provide a measure of the appropriate technique used in isolation.

7.1.1. Centralised Networks

Section 5.3.1.3 deals briefly with a number of algorithms which combine and extend the Kruskal, Prim, Esau-Williams, etc. algorithms for centralised network design. A summary of recorded results from experiments with these techniques is given in this section. All were obtained using an IBM PC-XT.

7.1.1.1. Two Pass Method

A method was suggested in section 5.3.1.3.1 for applying various established techniques on a partial solution in which some of the links incident with the centre have already been determined by calculating the unconstrained MST. The following table summarises the average improvement, in each case, given by the two pass method over the simple method, for different numbers of nodes.

Number of nodes :	25	50	75
Kruskal's Algorithm	0.6%	0.9%	1.3%
Prim's Algorithm	0.5%	0.6%	0.8%
Esau-Williams Algorithm	0.2%	0.4%	0.5%

Each entry is the mean figure for 100 tests. The improvements are not large but are clearly increasing as the number of nodes needing to be connected to the centre increases.

7.1.1.2. Hybrid Algorithm

The algorithm of section 5.3.1.3.2 involves the combination of the Kruskal [1956], Prim [1957] and Esau-Williams [1966] methods of centralised network design. At each stage of the algorithm, a modification of each type is considered. The following table summarises the average improvement of this method over each of the individual methods for different numbers of nodes.

Number of nodes :	25	50	75
Kruskal's Algorithm	0.7%	1.3%	1.5%
Prim's Algorithm	0.7%	1.2%	1.6%
Esau-Williams Algorithm	0.2%	0.3%	0.5%

Each entry is the mean figure for 100 tests. Again, the improvements are quite small but still increasing as the number of nodes increases.

7.1.2. Centralised Networks with Concentrators

Section 5.4.1.3 proposes a technique in which the Add and Drop algorithms of Bahl & Tang [1972] were combined in such a way that all possible additions and deletions were considered at each stage, the initial solution being a network with half the possible number of concentrators open. The average improvement given by this extended method over the superior of the Add and Drop algorithms, in each case, for different numbers of nodes, is as follows.

Number of nodes	% improvement
25	1.1
50	1.5
75	2.3

Each entry is the mean figure from 100 tests. The set of valid concentrator sites, in each case, was the set of node locations and the concentrator capacity ranged uniformly from 3 to the number of nodes. As with the previous two sets of figures, the improvements are quite small but become larger as the number of nodes increases.

7.1.3. The Double Drop Method

In section 5.5.2 the first practical approach to the NOP was encountered (the double drop method). This was a multi-stage, two-step algorithm in which, at each stage, all tandems were considered for removal, one at a time, with the optimum connection topology for each being approximated by a link dropping method operating in the same style.

Experimentation with this technique proves difficult. The only sensible comparison to make is between the solution produced by the double drop method with the known optimum for a number of examples. However, obtaining the optimum requires that an exhaustive search algorithm be applied to the networks under consideration. Whilst such an algorithm is easy to produce, it becomes impossible to run, even on the PRIME 9950 for n (the number of nodes) above about thirty.

In order to make even these smaller tests possible, it is necessary to place some restriction on the solution to the NOP in question. By restricting the degree of rerouteing to one extra tandem (between any communicating tandem pair) a large number of possible routeing strategies can be discarded.

The following table gives the average error produced by the double drop method with respect to the optimum solution for a number of different values of n. The DOR, in each case, is limited as suggested above.

Number of nodes	% error
5	0.0
10	0.2
15	0.8
20	1.7
25	2.0
30	3.7

Both sets of experiments were carried out using the PRIME 9950 computer. For n = 5, 10, 15 and 20, thirty tests were carried out. For n = 25 and 30, however, only ten and five respectively were possible due to the time taken for the exhaustive search algorithm to run with this number of nodes. For n=30, for example, the program must be run for 12 hours per night (to avoid annoying other users) for about a week in each case.

Limited as they are, however, these figures do suggest a pattern. The error is clearly increasing as the number of nodes increases and, it would appear, at an accelerating pace. It is only reasonable to speculate whether this trend levels off at a certain point or continues upwards past 100%, 200%, etc. where an error of 100a% implies that the double drop algorithm produces a solution costing a+1 times as much as the optimum.

Intuitively, it would seem that the error should level off at some point. For a comparable problem (the TSP), however, Sahni & Gonzalez [1976] have shown that this is not the case. It is shown that even the problem of finding a solution to the TSP which is guaranteed to be within any real constant b times the cost of the optimum is NP-complete.

In addition the scope for error apparently increases without bound as n increases. Suppose, for simplicity, that the optimum solution to the NOP in each case is a tree of some form. This tree has n-1 links in total (including tandem-tandem and node-tandem links) whereas the the original full mesh has n(n-1)/2. The probable number of links removed (and consequently the number of stages through which the algorithm passes) is thus quadratic in n.

In essence, the double drop method, despite being about the only practical method, to date, of dealing with the NOP would appear not to be recommended for larger networks.

7.1.4. Dealing with Greenfield Sites

It was noted in section 6.1.2.5 that the q replacement nodes which result from a reduction process acting on an original set of n nodes will, in general, be located at greenfield sites (i.e. sites that do not correspond to original node positions). Since a practical solution will require tandems to be chosen from the existing node sites, it will be necessary to move these locations at some stage.

As stated in section 6.1.2.5, there are two obvious ways of doing this: the q nodes at greenfield sites can be moved to real sites before tandems are chosen or a set of greenfield tandems can be chosen from the q greenfield nodes and these tandems then moved to real sites.

10 tests were performed with n (the number of nodes) varying from 30 to 100, q varying from 15 to 30 and M (the number of tandems) varying from 1 to 5. In each case the network was reduced from n nodes to q before, on the one hand, moving the nodes and then choosing the tandems (by a star tandem network assumption followed by perturbation) and, on the other, by choosing the tandems and then moving them. In all 10 cases, the final set of chosen tandems was identical in each case. The IBM PC-XT was used for this set of experiments

7.1.5. Errors in Reduction

The reduction stage of the compound algorithm presented in chapter 6 is, in many ways, the most important single component. With a reduced set of nodes established, the subsequent stages of the compound algorithm will, in general act upon a similar number of nodes and tandems in each case. The reduction process, however, has the power to represent any conceivable network of several hundred nodes in this standard form.

In addition, while comparison of compound results with exact solutions is possible for small numbers of nodes, it is not for larger numbers. If the results for small values of are n good (which they are - see section 7.2) then the only problem with extending the algorithm to larger vales can be the possible inaccuracy of the reduction process. If the accuracy of this method can be verified, there is then some considerable justification for claiming the compound algorithm to be effective (in general) for all sizes of network. With this in mind, a large amount of experimentation has been carried with a view to investigating the accuracy (or otherwise) of this section.

The single source of error in the reduction process is that a node i may, during the course of the process, be combined into a group J, the replacement node for which is further away from the node i than is another replacement node for a group K. If i is to be connected to its nearest tandem and both replacement nodes J and K eventually become tandems then i will be connected to K instead of J. The tandem K will not, however, be positioned in the ideal location to serve its group since its position was determined without any consideration for the node i (it is at the weighted centre of the group of nodes it has replaced). Tandem J will be located incorrectly for a similar reason. The experiments must therefore take the form of a series of comparisons between sets of nodes produced by the reduction process and sets of nodes determined on the basis of their distance from the replacement nodes.

The PRIME 9950 was used for all experiments in this section.

7.1.5.1. Number of Nodes in Error

The basis of the following test results is very simple. In each case, a network was generated at random with the weight of each node uniformly distributed within a fixed interval ([1,10] in fact but this is academic - only the ratios are of importance). The starting set of n nodes, i=1,2,..,n, was replaced, via the normal weighted reduction algorithm by a set of q nodes, j=1,2,..,q. A check was kept, at each stage, of the combination of node pairs so that each node i' could, on termination of the reduction process, be assigned to a particular group j'. For such a node i', the distance from i' to all of the other replacement nodes j=1,2,..,q was calculated. If a replacement node k was found such that the distance from i' to k was shorter than from i' to j' then the node i' was said to be error as a result of the reduction process. The following table summarises the results of a large number of experiments of this sort.

n	q	Number of tests	Number of nodes in error in each case	Mean number of nodes in error	Mean % of nodes in error
100	10	60	3,3,2,3,2,3,2,2, 3,4,1,2,2,3,2,5, 3,2,3,1,2,2,2,2, 3,2,1,3,3,1,4,5,1, 2,3,3,4,3,3,3,1,1, 1,2,3,1,2,4,7,2,3, 4,3,0,6,4,1,1,2,3	2.6	2.6

100	30	60	1,1,2,0,0,0,1,0, 0,0,2,1,0,1,1,1, 0,0,0,0,1,2,1,0, 0,1,0,1,2,1,0,1, 1,0,1,1,1,0,1,1, 1,0,0,0,0,0,0,2, 0,1,3,1,0,1,0,1, 0,0,3,0	0.7	0.7
100	50	40	0,1,0,0,1,0,0,0, 0,0,0,1,0,0,0,1, 0,0,2,0,0,2,0,0, 0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0	0.2	0.2
200	50	30	3,2,2,3,2,3,1,2, 1,1,1,1,2,2,1,3, 2,2,2,2,0,1,1,2, 2,4,3,3,2,2	1.9	1.0
300	50	10	10,7,2,1,4,8,9, 4,5,5	5.5	1.8
500	50	10	17,15,18,13,15, 10,17,19,20,11	15.5	3.1

A clear, and not entirely unexpected, pattern emerges. In general, the errors will increase with larger values of n and smaller values of q (since each implies that more reduction has to take place). The mean value given in each case is, in effect, the value of E (the expected number of nodes in error) in section 6.1.3.1.2. P (the probability of at least one node in error), on the other hand, is given by the ratio of the number of times the count was non-zero to the total number of tests.

7.1.5.2. Shape of the $P_{n'}$ Curve

The previous section presents numerical evidence of the number of nodes typically in error for a particular pair of values for n and q. It is also of considerable importance to know the stage within the process at which these errors occur. If there is a tendency for most to occur toward the end of the reduction then a small (but nevertheless time consuming) increase in q may result in a good increase in accuracy. This, however, appears not to be the case.

Fig 7.a shows a number of typical examples. These were obtained by a two pass method somewhat comparable method to above. Networks of 100 nodes were reduced to 10, 30 and 50 replacement nodes respectively. Those nodes found to be in error were recorded. An identical reduction process was then carried out on the original set of nodes. The marked nodes were monitored to determine the stage at which they left the correct group (i.e. the stage at which they combined with a node from a different group). Fig 7.a shows the results of 140 experiments for each of q=10, 30 and 50. The error curve represents the number of times (out of 140 tests) that an error was recorded at the n'th stage. Dividing the magnitude of the error curve at each stage by the number of tests (140 in this case) will give the $P_{n'}$ curve described in section 6.1.3.1.2.

7.1.5.3. Discussion of Reduction Results

Experiments suggest the number of errors within the reduction process to be small. The vast majority of the nodes are grouped in the same way as they would be if the positions of the replacement nodes were known initially and the nearest chosen in each case. These replacement nodes therefore do represent the original network well, This must be greatly appreciated since, if the latter stages of the compound algorithm are to be of any use, the problem being dealt with must at least be an appropriate one.

Of course even where there are nodes in error, this in no way automatically implies that there will be an error in the final solution. It is probable that the relatively small discrepancy caused

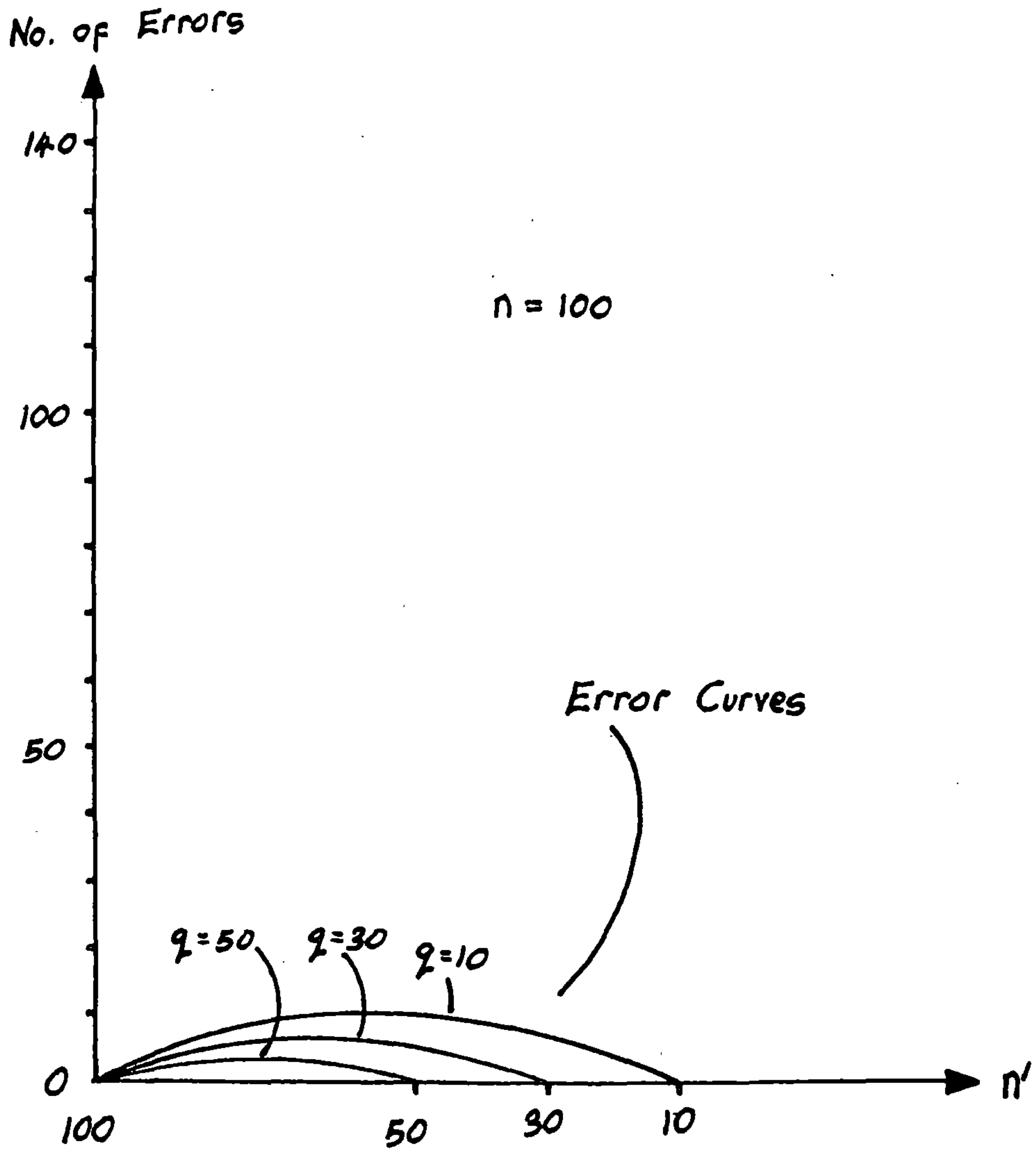


Figure 7.a. Reduction Errors at Each Stage

by the misplacing of a node will have no effect on the eventual outcome in global terms.

Suppose a node is placed in an incorrect group. This implies two of the replacement nodes are a small distance from their correct location and that their traffic values are slightly high or low. The eventual effect of this may be nullified for a number of reasons.

1. The replacement nodes, placed at greenfield sites by the reduction process, are moved to the closest actual site once reduction has finished. Bearing in mind the small error in placement, it is likely that the same node is chosen as would have been if no errors has occurred.

2. The process of choosing tandems is global and is unlikely to be affected by small local changes. Whether a node represents a good tandem depends upon its location with respect to the others. Distance errors on a local scale will be very small compared to inter-tandem distances.

3. In the unlikely event of an incorrect tandem being chosen despite points 1 and 2, there is still the perturbation process to come which may generate the superior solution.

The shape of the $P_{n'}$ curve (Fig 7.a) is interesting. It clear from experiment that the probability of a error is greatest midway through the reduction process. Fewer errors occur at the start and finish. An explanation of this behaviour becomes available when the node replacements are considered stage by stage. At each stage n' $(n \geq n' \geq q+1)$ two nodes, a distance $d(n')$ apart, are combined to form one. A error will occur when these two nodes are either side of a boundary between two of the eventual groups.

At the early stages of reduction the nodes are mainly separate and the clustering process has not progressed far. In this respect, errors are more likely since the eventual groups are a long way from being established. Towards the end of the process the groups are better established and the probability decreases.

On the other hand, however, the distance d(n') between the appropriate two nodes increases as the process progresses. The probability of the two nodes spanning a boundary therefore increase as well. In this sense errors become more likely as the number of nodes in the network becomes smaller.

Combining these two observations indicates that while errors are unlikely at the start of the reduction process, due to the small distances involved, and at the end, due to the groups being well defined, neither of these features is dominant in the middle.

The small probability of error in the closing stages has an important implication, namely that the precise value of q would not appear to be crucial. q can be made small enough to allow the subsequent optimisation routines to run comfortably.

7.2. Experiments Involving Complete Networks

This section is concerned with reporting the results of experiments on the compound algorithm of section 6.5. To achieve this, a number of complete network examples are acted upon by this compound algorithm and the solutions compared with those obtained from an exhaustive search program running on a large machine.

7.2.1. Outline of Experimentation Process

The principles involved in practical experimentation are quite simple. A number of example network problems were collected. Some of these were real world problems while others were constructed artificially for the purposes of testing (the number of actual problem networks available is obviously limited). Comprehensive comparisons were then made between the solution obtained from the compound algorithm and by certain other methods. A study of the running times is also necessary. A wide range of costs were used in the test process (Upton [1985]), producing the following results, in order to test the sensitivity of the compound method.

7.2.2. Results of Experiments

The major part of the testing process for the compound algorithm takes the form of a series of comparisons between the solution produced by the algorithm and the known optimum produced by an exhaustive search approach. Clearly such comparisons can only be made on limited numbers of nodes. Sections 7.2.2.2 and 7.2.2.3 deal with extended testing involving larger networks.

7.2.2.1. Optimum Comparison

The results of 15 experiments are contained in this section (5 real network problems - 10 fabricated network problems). Each problem was acted upon in two ways.

1. An exhaustive search program of the type described in section 5.2. This program runs on the PRIME 9950 and is guaranteed to find the optimum.

2. The compound algorithm developed in chapter 6, running on the IBM PC-XT, with reduction (if necessary), star optimisation and various degrees of perturbation and lookahead.

The usual approach was to begin with the compound algorithm in its simplest form (i.e. $x=y=z=1$ for perturbation and $r=1$ for lookahead - see section 6.5.1.6) and compare results with the known optimum. If the results matched, there was no need to go any further: if not, the values of x, y, z, and/or r were varied and the experiment repeated. The following table presents the results obtained. $x=y=z=r=1$ unless otherwise stated. The underlined figure represents the best solution (with respect to the number of tandems) in each case.

Experiment network number	Number of nodes	Number of tandems	Cost of solution produced by PRIME (£M)	Cost of solution produced by IBM PC-XT (£M)	Percentage error of compound algorithm
1	15	1	1.010	1.010	0.0
		2	0.923	0.923	0.0
		3	<u>0.892</u>	<u>0.892</u>	0.0
		4	0.897	0.897	0.0
		5	1.114	1.114	0.0
2	16	1	1.552	1.552	0.0
		2	1.310	1.310	0.0
		3	<u>1.303</u>	<u>1.303</u>	0.0
		4	1.411	1.421	0.7
2 (x=y=2)		4	1.411	1.411	0.0
		5	1.606	1.613	0.4
2 (r=2)		5	1.606	1.606	0.0
3	17	1	0.883	0.883	0.0
		2	<u>0.729</u>	<u>0.729</u>	0.0
		3	0.757	0.757	0.0
		4	0.802	0.802	0.0
		5	0.916	0.916	
4	18	1	1.512	1.512	0.0
		2	<u>1.484</u>	<u>1.484</u>	0.0
		3	1.509	1.509	0.0
		4	1.562	1.562	0.0
5	19	1	1.997	1.997	0.0
		2	1.603	1.603	0.0
		3	1.591	1.591	0.0
		4	<u>1.590</u>	<u>1.590</u>	0.0
		5	1.668	1.670	0.1

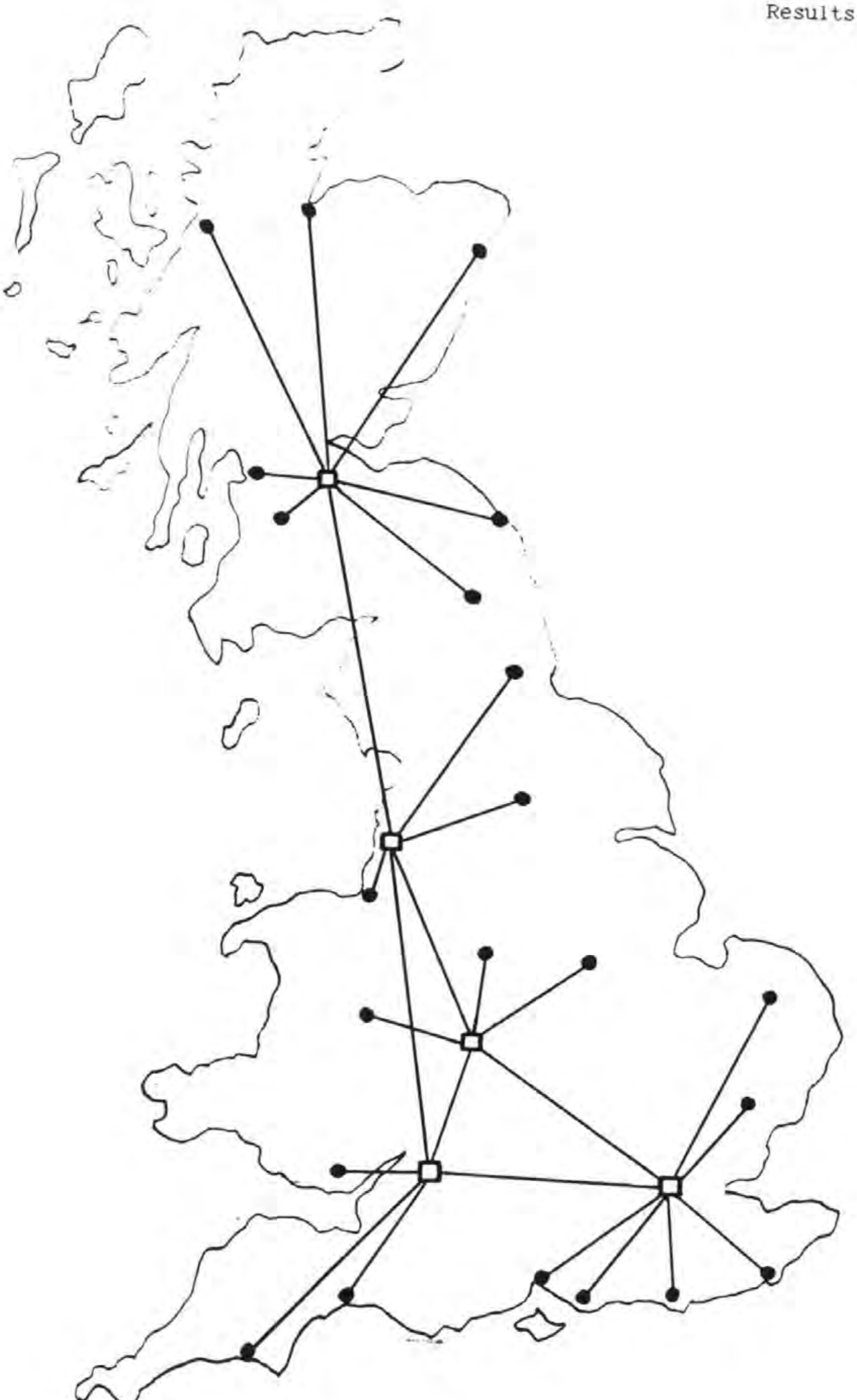
5 (r=2)		5	1.668	1.668	0.0
6	20	1	2.398	2.398	0.0
		2	2.310	2.310	0.0
		3	2.247	2.247	0.0
		4	2.299	2.299	0.0
7	21	1	1.935	1.935	0.0
		2	1.895	1.895	0.0
		3	1.891	1.891	0.0
		4	1.899	1.908	0.4
7 (r=2)		4	1.899	1.908	0.4
7 (r=3)		4	1.899	1.899	0.0
8	22	1	1.750	1.750	0.0
		2	1.511	1.511	0.0
		3	1.473	1.473	0.0
		4	1.489	1.489	0.0
9	23	1	2.421	2.421	0.0
		2	2.003	2.003	0.0
		3	1.832	1.832	0.0
		4	1.818	1.818	0.0
		5	1.989	2.010	1.1
9 (r=2)		5	1.989	1.989	0.0
10	24	1	1.133	1.133	0.0
		2	1.061	1.061	0.0
		3	1.096	1.096	0.0
		4	1.272	1.272	0.0
11	25	1	0.965	0.965	0.0
		2	0.952	0.952	0.0
		3	1.001	1.001	0.0
		4	1.090	1.122	2.9
11 (r=2)		4	1.090	1.090	0.0

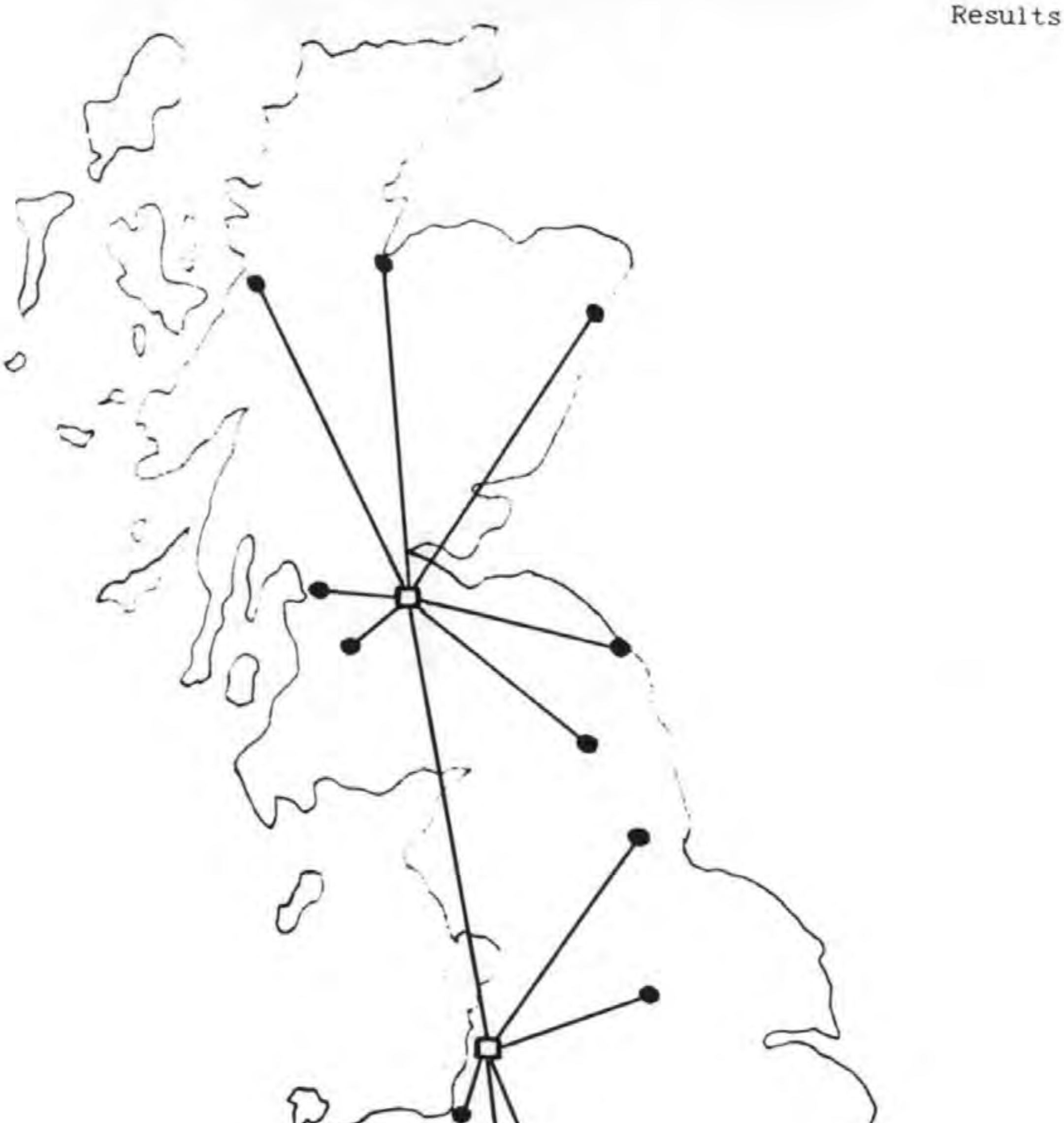
12	26	1	2.225	2.225	0.0
		2	2.058	2.058	0.0
		3	1.981	1.981	0.0
		4	2.002	2.002	0.0
13	27	1	1.715	1.715	0.0
		2	1.661	1.661	0.0
		3	1.688	1.688	0.0
		4	1.849	1.849	0.0
14	28	1	1.406	1.406	0.0
		2	1.335	1.335	0.0
		3	1.218	1.218	0.0
		4	1.442	1.451	0.6
14 (r=2)		4	1.442	1.451	0.6
14 (r=3)		4	1.442	1.442	0.0
15	29	1	2.623	2.623	0.0
		2	2.051	2.051	0.0
		3	1.950	1.950	0.0
		4	2.033	2.058	1.2
15 (r=2)		4	2.033	2.033	0.0

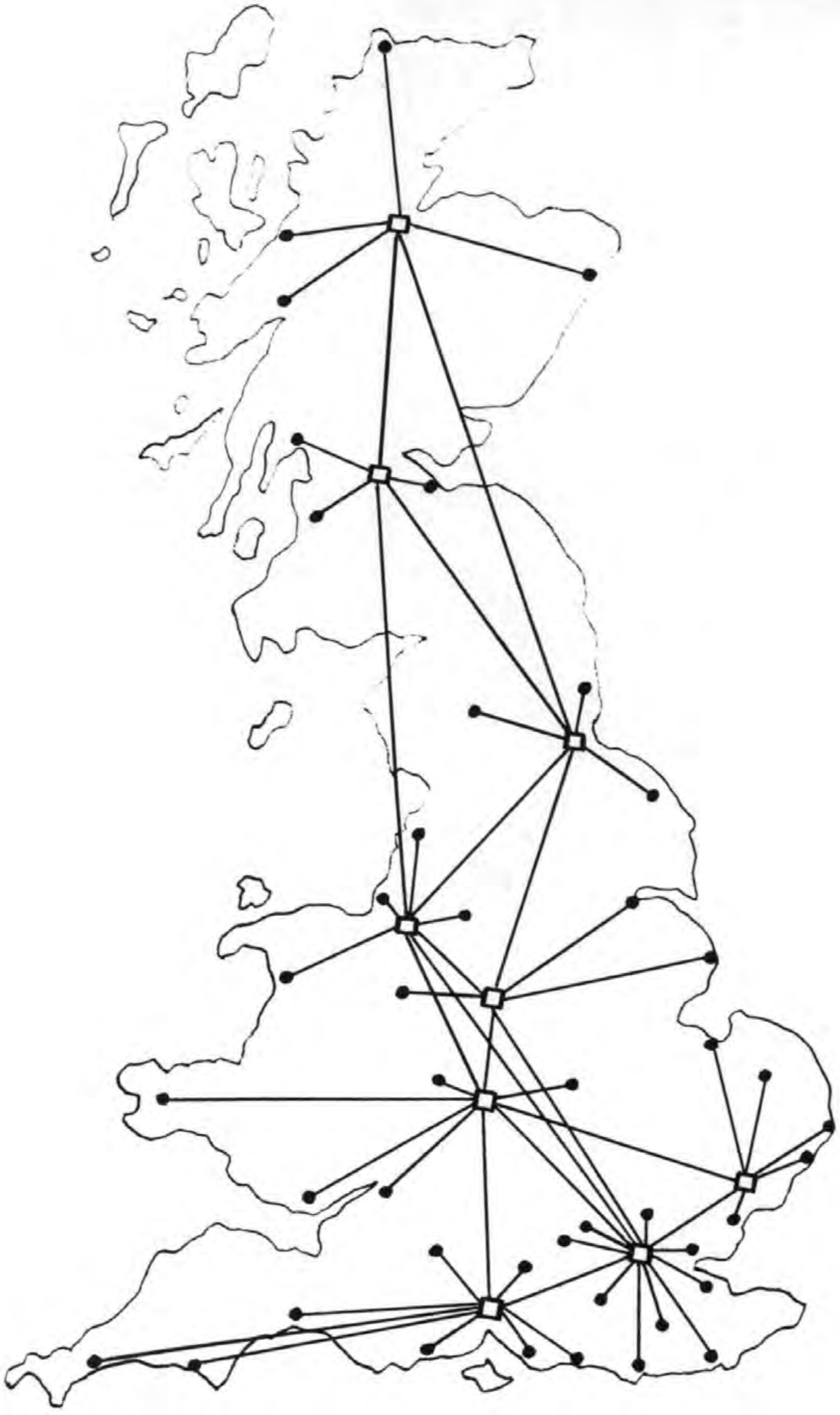
This set of test networks represents the largest (in terms of numbers of nodes) that can be compared in this way. As the number of nodes approaches 30 the exhaustive search method takes up weeks of CPU time particularly as the tests were extended beyond the optimum number of tandems. The implications of these results are discussed in section 7.2.3.

7.2.2.2. Symmetrical Testing

The problem of the inability to test large networks is a basically insoluble one. However some useful results can be obtained for medium or large networks as described briefly in this section and the next.







Three perfectly symmetrical networks were generated with 37, 49, and 50 nodes respectively. It is not possible to find the exact optimum with complete certainty but the structure of the networks along with the distribution of traffic was such that the locations of the tandems, at least, could be guessed with some degree of confidence (for example the 37 node network of Fig 7.b.

In all three cases the set of selected tandems was the expected set although perturbation with x=y=2 was necessary for the 49 node network (the solution being 0.8% in error with x=y=1 - see Grout, Sanders & Stockel [1987a & 1988]). As for the tandem interconnection, no superior solution could be found by hand to the one generated by the compound algorithm in each experiment.

7.2.2.3. Double Drop Comparison

It is impossible to test large networks by exhaustive search. The idea of using predictable symmetrical networks is less effective also since it is far more difficult to guess the solution accurately. Under these circumstances, a new approach is necessary.

Rather than compare the solution produced by the compound algorithm with the known optimum, it will be matched against the solution from the double drop algorithm which is the most effective and practical method to date. It is known however that the double drop algorithm tends to produce solutions which diverge from the optimum as the number of nodes increases.

The details of two large existing networks were available as data. The first had 132 nodes and the second 156. For the 132 node network the double drop algorithm produced a solution involving 4 tandems. The compound algorithm gave a solution with 3 tandems with a saving of 6.5%.

The solutions for the 156 node network were identical for both methods.

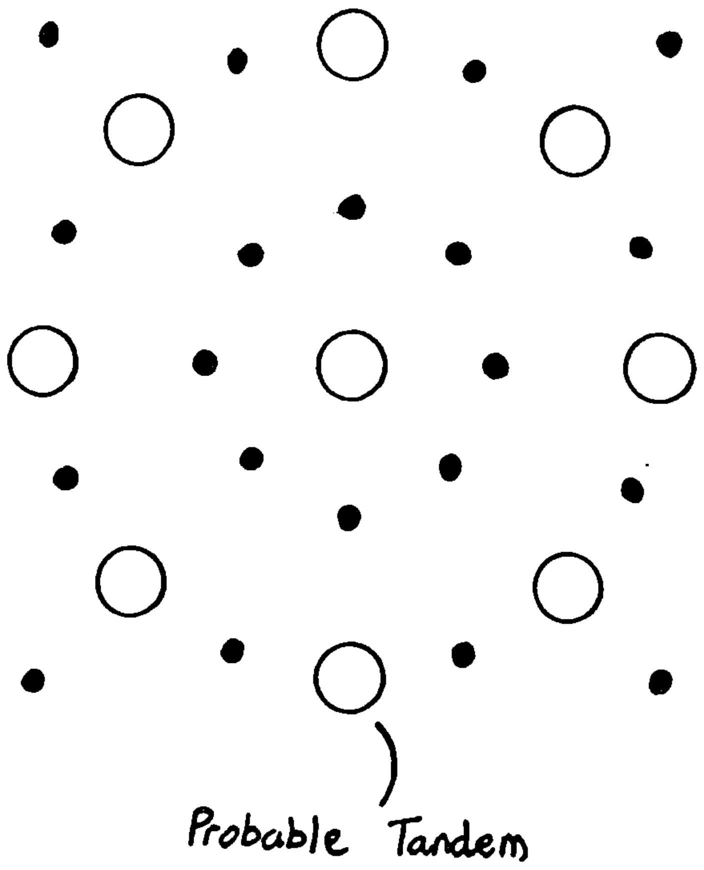


Figure 7.b. 37 Node Symmetrical Network

7.2.2.4. Running Time

It is impossible to specify exact running times for the compound algorithm for a network of any given size. For a network of n nodes, the complexity of the reduction and initial tandem selection sections can be determined accurately since their operation remains constant under all circumstances. The tandem perturbation and lookahead optimisation section, however, will both run as long as improvements to the current solutions are being found and this is a flexible area of the algorithm. Sections 7.2.2.4.1-5 deal with each of the five main sections of the compound algorithm separately. The times given are all measured on the IBM PC-XT operating with no co-processor etc. Clearly such speeds can be improved by orders of magnitude on superior machines.

7.2.2.4.1. Reduction

The reduction process reduces a network of n nodes down to a network of q nodes. As seen in section 6.1.3.3 the complexity of this operation is bounded by $O(n^3)$ steps. Clearly the observed running time will depend on the values of q and n. Some approximate examples of recorded times are as follows.

From n = 50 to q = 5, time = 2 minutes
From n = 100 to q = 30, time = 10 minutes
From n = 200 to q = 50, time = 30 minutes

In fact the value of q matters little in practical situations since the reduction is much quicker in the latter stages (less comparisons to make between pairs of nodes)

7.2.2.4.2. Star Tandem Network Optimisation

This sections selects an initial set of tandems by assuming a star connection topology. Its running time will depend upon the number of tandems actually selected. If this is known (which it clearly is afterwards) the running times can be stated confidently. The following table gives the running time for each value of M (the number of tandems) as well as the values of q_M (the reduced number of nodes in the algorithm of section 6.2.4.1) used to achieve this.

M	q_M	Running time (minutes)
1	50	< 1
2	40	1
3	30	5
4	30	10
5	28	15
6	22	20
7	20	20
8	19	20
9	19	25
10	19	30

The values of q_M are deliberately chosen to ensure the times remain within acceptable limits.

7.2.2.4.3. Tandem Perturbation

This section acts upon the solution of the previous section with a series of perturbations. Its running time is entirely variable depending upon the number of times an improvement is found which in turn depends upon the complexity of the perturbations applied. The following table gives some typical observed examples for M tandems and q_M nodes with various amounts of perturbation.

q_M	M	Observed times
30	4	10 minutes, 20 minutes, 25 minutes
50	1	5 minutes (maximum)
19	10	30 minutes, 1 hour, 2 hours
40	2	10 minutes (maximum)

Choosing the values of q_M wisely can have important benefits in this section as well.

7.2.2.4.4. Lookahead Tandem Network Optimisation

This section (which determines the tandem network connection topology via a lookahead series of link removals) is similar to the previous one in that its running time will depend on how many such removals are made. Timings may be made considerably more accurately in this case, however, due to the tendency for the vast majority of networks to have similar structures (tending towards a tree but with a few extra circuits). The following table gives the approximate times for different values of M and r (the number of lookahead stages).

M	r=1	r=2	r=3	r=4
1	No optimisation necessary			
2	No optimisation necessary			
3	1 sec	Unnecessary......................		
4	2 secs	15 secs	2 mins	Unnecessary..
5	1 min	20 min	1 day	Not practical
6	5 min	1 hour	>1 day	Not practical
7	15 min	4 hours	Not practical	
8	35 min	20 hours	Not practical	
9	2 hours	Not practical		
10	5 hours	Not practical		

If the number of tandems is large (which is fortunately rare) this can be the most complex part of the compound algorithm.

7.2.2.4.5. Local Network Optimisation

This section merely determines which of the available types of transmission link is most suitable for each node-tandem link by a series of cost comparisons. It takes no more than a few seconds even for several hundred nodes.

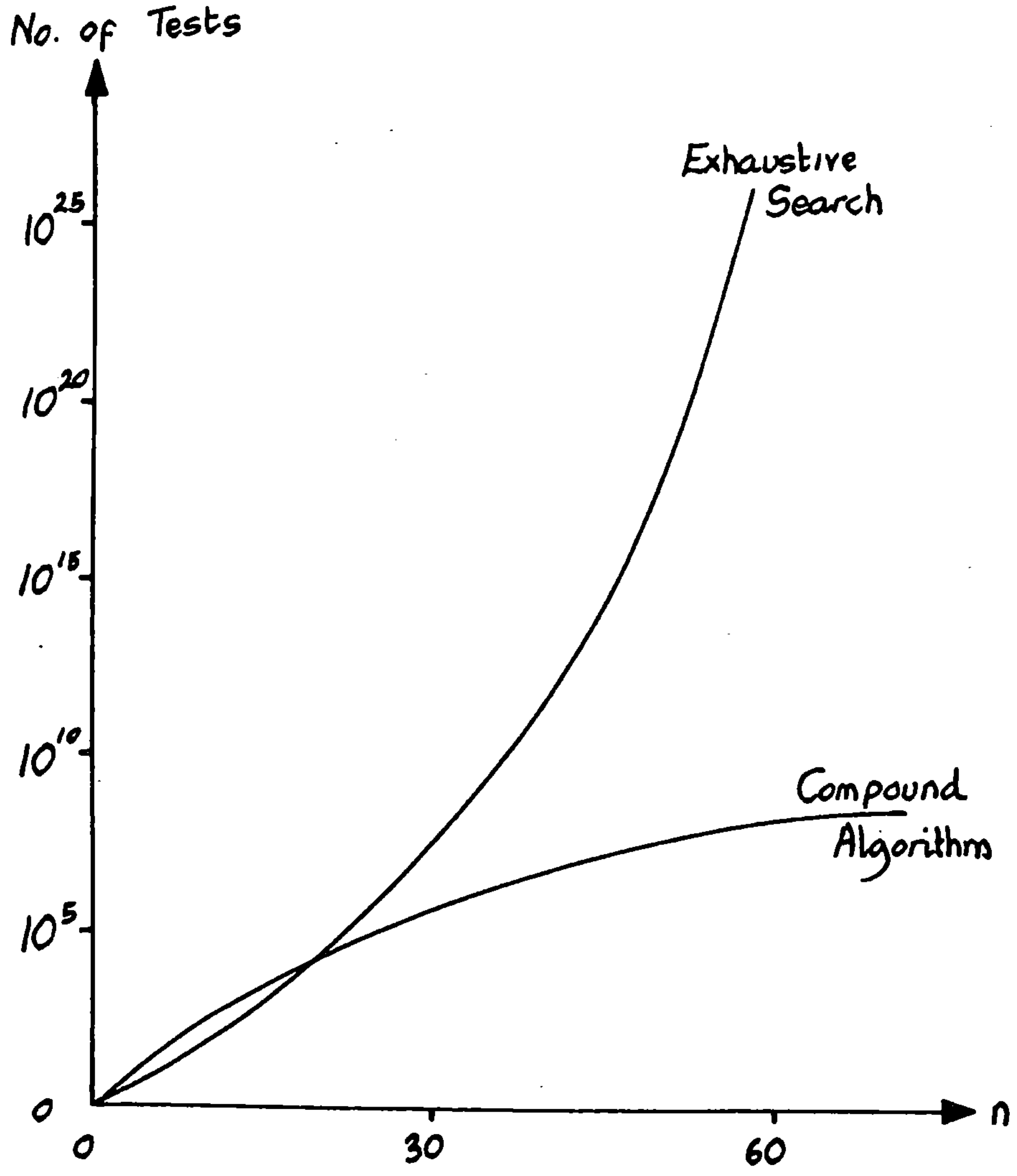


Figure 7.c. Running Time of Compound Algorithm

7.2.2.4.6. Combined Running Time.

Fig 7.c shows the running time of the compound algorithm against that of an exhaustive search method. Whilst the running time of exhaustive search algorithm increases exponentially with the number of nodes (n), the compound algorithm is effectively bounded since any network may be reduced to the same value of q nodes. Any reduced network where the number of nodes is bounded can be dealt with within a time bounded by a constant (albeit a large one). The running time of the reduction process, on the other hand increases only as a polynomial.

7.2.3. Discussion of Results

For relatively small (<30) numbers of nodes the results contained in this chapter are excellent. In many cases the optimum can be found using the simplest version of the compound algorithm (i.e. with the lowest level of perturbation and single stage lookahead). In the few cases where this is not sufficient, a minor extension of either the perturbation section or the optimisation process provides the necessary extra sophistication.

These small networks represent the limit of the practical usage of exhaustive search comparison techniques since the exact optimum cannot be determined, with certainty, for larger ones.

Moderately sized symmetrical networks can be used in which the exact optimum can be guessed with confidence. The results appear perfect for these examples but it may well be that these networks are particularly easy for the compound algorithm to deal with. There is no immediate reason why this should be so but of course they were chosen in the first place because their solution was intuitively obvious.

Even this approach fails for particularly large networks and comparisons with the double drop method are necessary for hundreds of nodes. The experiments performed suggest the compound algorithm to be superior to the double drop method for these large networks.

A probably better vindication of the compound algorithm is available however. The method has been shown to be a good one for small networks and the only section of the compound algorithm which comes into play for large networks but not for small ones is the reduction process. This process is deliberately constructed so that, if functioning correctly, it will produce a replacement network which accurately represents the original. The only problem that can arise is that the process does not function correctly, i.e. that serious errors are made within the reduction. Section 7.1.5.3, however, suggests clearly that this is not the case.

Running times in the order of hours may appear unacceptable. A small amount of consideration, however, puts a different perspective on matters. Firstly the compound algorithm could run up to 1000 times faster on more powerful machines (Sumner [1986]), thus reducing quite accurate optimisation to seconds and minutes. Secondly the huge cost of the network under consideration will usually justify the most elaborate means of optimisation.

In summary, the compound algorithm provides excellent solutions for small networks and there is no reason to assume that this will not remain the case for larger networks. Even more importantly, it provides a method for dealing with these networks where, in all practicality, none previously existed.

CHAPTER 8

EXTENSIONS TO OTHER AREAS OF OPTIMISATION

Much hath been done, but more remains to do-
Their galleys blaze- why not their city too?
George Gordon, Lord Byron
The Corsair, 1814

The techniques of reduction, perturbation and lookahead were presented in chapter 6 both in general theoretical terms and within the context of solving various aspects of the NOP. In some case, the descriptions were limited to a level necessary for such purposes.

The underlying principles, however, can be applied to a number of other areas of problem solving and optimisation in particular. This chapter has two objectives - to demonstrate the generality of the techniques and to discuss a few particular applications in slightly greater detail.

8.1. Reduction Techniques

A large n node network can be represented by a smaller q node replacement network by the process of step by step reduction. The principle of replacing two (or more) objects by a single object which represents the characteristics of the original two, however, can be extended far beyond this particular application. Any large collection of objects can be represented by a smaller set in this way. Certain types of optimisation or problem solving can then be effected upon this reduced set. Typical examples are as follows.

1. Selecting a subset of the original set of objects for any reason (as with selecting tandems from nodes).

2. Characterising the original set of objects.

3. Comparing two sets of objects for similarity.

These operations can all be carried out with the reduced set of objects instead of the original set. The objects in question may be almost anything in principle. For example, the records

NAME	: Peter Smith		NAME	:	Paul Dupont
NATIONALITY	: British	and	NATIONALITY	:	French
AGE	: 37		AGE	:	31
HEIGHT	: 5'10"		HEIGHT	:	6'00"

within a data file, could be replaced (say) by the record

NAME	: Replacement 001
NATIONALITY	: European
AGE	: 34
HEIGHT	: 5'11"

The complete generality of this concept clearly presents a problem, namely the representation of non-numerical data in the replacement object (here the combination of British and French to form European seems reasonable but this will not always be the case). Equally the ages and heights might not be replaced by a single figure, in each case, but by a range of values, say 31-37 and 5'10"-6'00" in the example above. The next section defines the necessary rules in more precise terms. Section 8.1.2 describes one particular application.

8.1.1. Generalised Reduction

A <u>generalised reduction technique of order h</u> will act upon an initial set of n <u>objects</u>, $S_n=\{I_1,I_2,..,I_n\}$, transforming the initial set, via a sequence of intermediate sets, $S_p=\{O_1,O_2,..,O_p\}$, to a final set of q objects, $S_q=\{F_1,F_2,..,F_q\}$. The process consists of a number of stages. Each stage reduces an intermediate set of p objects to an intermediate set of p-h+1 objects. Some modification will be necessary at the last stage if n-q is not an integer multiple of h-1. This will generally take the form of a single stage of order p-q+1 reduction where p is the number of objects in the penultimate set. This simple requirement is not considered here.

An object O_i, <u>of degree d</u>, consists of d <u>characteristics</u>, c_{11}, c_{12}, ..., c_{1d}. The reduction process can only act upon a set of objects of equal degree. There are two components to each stage of the reduction process.

8.1.1.1. Selection Stage

For each k=1,2,...,d, define a <u>characteristic distance measure of order h</u> f_k in such a way that $f_k(c_{1k}, c_{2k}, \ldots, c_{hk})$ gives a numerical measure of the dissimilarity between the kth characteristic of objects O_1, O_2, ..., O_h (see section 8.1.1.3). Define the <u>generalised distance</u> between the h objects O_1, O_2, .., O_h, of degree d, to be

$$D(O_1,O_2,\ldots,O_h) = F(f_1(c_{11},c_{21},\ldots,c_{h1}),f_2(c_{12},c_{22},\ldots,c_{h2}),\ldots\ldots,f_d(c_{1d},c_{2d},\ldots,c_{hd}))$$

where F is some function of the individual characteristic distances. Objects $O_{1'}$, $O_{2'}$, ..., $O_{h'}$ are selected such that $D(O_{1'},O_{2'},\ldots,O_{h'})$ is a minimum. These objects will be replaced by a single object at the replacement stage.

As an illustration, for the network reduction process, h=2, d=3, $c_{i1}=x_i$, $c_{i2}=y_i$ and $c_{i3}=w_i$. The characteristic distance measures are given by $f_1(c_{i1},c_{j1})=x_i-x_j$, $f_2(c_{i2},c_{j2})=y_i-y_j$ and $f_3=0$ (irrelevant). The distance between two nodes is given by the euclidean form of $F(i,j)=\surd[f^2{}_1(c_{i1},c_{j1})+f^2{}_2(c_{i2},c_{j2})]$.

8.1.1.2. Replacement Stage

The h objects $O_{1'}$, $O_{2'}$, ..., $O_{h'}$ chosen at the selection stage are replaced by an object O_r defined by characteristics c_{r1}, c_{r2}, ..., c_{rh} where each c_{rk} is given by $c_{rk}=g_k(c_{1'1},c_{1'2},\ldots,c_{1'd},c_{2'1},c_{2'2},\ldots,c_{2'd},$

.. ,$c_{h'1}$,$c_{h'2}$,...,$c_{h'd}$), a function of (possibly) all characteristics of each of the replaced objects.

For the network reduction process, $O_{1'}$=i', $O_{2'}$=j', $c_{r1}=x_r=(w_{i'}x_{i'}+w_{j'}x_{j'})/(w_{i'}+w_{j'})$, $c_{r2}=y_r=(w_{i'}y_{i'}+w_{j'}y_{j'})/(w_{i'}+w_{j'})$ and $c_{r3}=w_r=w_{i'}+w_{j'}$.

8.1.1.3. Dealing with Qualitative Data

These generalised techniques assume that each characteristic of the objects is a numeric value. In some circumstances, this may not be the case. If some qualitative value is used it is essential that it is possible to represent this quantity on a numeric scale. Often this is quite simple. For example, the weather can be graded from 0 (raining all day) to 100 (sunny all day) with various definitions in between. Equally, any numeric value within the correct limits must have a corresponding qualitative equivalent.

Sometimes, however, such a scaling is not possible, particularly where there is some element of multi-dimensionality in the data. In such cases, the only solution is use a look-up table system in which each replacement is explicitly recorded according to some agreed pattern. For example:

```
........................
REPLACE Footballer AND Tennis Player BY Ball Sport Player
........................
```

Similarly the distance measures between qualitative elements of data will need to be explicitly defined. For example:

```
....
f(Footballer,Rugby Player) = 1
f(Weightlifter,Skier) = 5
....
```

Any such system of distance measure and replacement, however, is likely to be extremely subjective and consequently lacking in rigour.

The technique of scaling qualitative data is also to be preferred to the look-up table approach for reasons of efficiency.

The study of even further generalised methods of reduction is the concern of cluster analysis (Anderberg [1973] and Spath [1980]), otherwise referred to as numerical taxonomy. A discussion of these techniques would be deviating from the intended direction of this chapter.

8.1.2. The Euclidean Travelling Salesman Problem

This section considers one area of optimisation in which reduction may be used. It is a brief summary of the work to be found in Grout, Sanders & Stockel [1987b].

The asymmetric travelling salesman problem (ATSP) is a classical example of an NP-complete combinatorial problem (Karp [1972]). Given n cities $i=1,2,..,n$, and a cost c_{ij} of travelling between (or connecting) each pair of cities i and j, it is required to find a cyclic permutation π of the cities (a tour) which minimises

$$z = \sum_{i=1}^{n} c_{i\pi(i)}$$

The Euclidean travelling salesman problem (ETSP) is a special case of the ATSP in which the planar distance conditions of section 2.1.4 are satisfied. It has also been shown to be NP-complete (Papadimitriou [1977]).

8.1.2.1. Assignment Problem Solutions and Patching Algorithms

The asymmetric assignment problem (AAP) requires that any (not necessarily cyclic) permutation σ of the cities be found which minimises z as defined above (with σ replacing π). The AAP can be solved in $O(n^3)$ steps (Kuhn [1955]) steps and gives, on average, large subtours in its solution (Grout, Sanders & Stockel [1987b]). This set of subtours provides a useful starting solution to a slight variation upon a perturbation method.

Suppose q subtours (the σ permutation) are found as the solution to the AAP on a given set of n nodes. An approximation to the solution for the corresponding ATSP (the π permutation) can be found as follows.

```
I={$x_1$,$y_1$,$x_2$,$y_2$,..$x_n$,$y_n$,$\sigma$,q} ;
q' := q ; $\pi$ := $\sigma$ ;
WHILE q'>1 DO
    BEGIN
    minimum := $\infty$ ;
    FOR i := 1 TO n DO
        FOR j := 1 TO n DO
            IF (i and j are in different subtours) THEN
               IF $c_{i\pi(j)}+c_{j\pi(i)}-c_{i\pi(i)}-c_{j\pi(j)}$<minimum THEN
                 BEGIN
                 minimum := $c_{i\pi(j)}+c_{j\pi(i)}-c_{i\pi(i)}-c_{j\pi(j)}$ ;
                 i' := i ;
                 j' := j ;
                 END ;
    q' := q'-1 ;
    temp := $\pi$(i') ;
    $\pi$(i') := $\pi$(j') ;
    $\pi$(j') := temp ;
    END ;
O={$\pi$} ;
```

This is known as a patching algorithm since the q subtours are systematically patched into one by combining, one stage at a time, the two subtours which minimise the increase in cost.

8.1.2.2. Initial Solutions via Reduction

There are certain characteristics of the Euclidean assignment problem (EAP) which have a detrimental effect on its suitability as an initial solution to a patching algorithm for the ETSP. There is a very strong tendency for such a solution to consist entirely of subtours containing only two or three cities (Grout, Sanders & Stockel [1987b]). This implies that any patching algorithm will be operating from an almost trivial initial solution, suggesting little improvement over the simple algorithm offered by Karg & Thompson [1964].

Fortunately, the particular form of the ETSP allows an alternative method of generating initial subtours. Since the cities are regarded as points in the Euclidean plane, they can be grouped in regions and subtours formed within each region. It will only be necessary to ensure that each region contains few enough nodes to allow the optimum subtour to be constructed.

These regions could be obtained by arbitrarily dividing the plane in a predetermined way (into a series of rectangles, say, as proposed by Karp [1976]). There are, however, problems with this approach.

1. The arbitrary nature of the choice of regions gives no guarantee that the groupings are close to the best possible.

2. There is no immediate way of ensuring that a large number of cities are not contained within the same region.

The underlying problem is that such a method of grouping is top-down whereas a bottom-up approach would appear to be superior. Reduction is such a method. Suppose the maximum number of cities permitted in any group is m. The following algorithm will then reduce the n cities down to q replacement cities (q being unknown at the beginning of the process).

RTSP1. Set $w_i=1$ for all $i=1,2,..,n$. Set $n'=n$.

RTSP2. Find i' and j' such that $c_{i',j'}=\min_{ij}(c_{ij})$.

RTSP3. If $w_{i'}+w_{j'}>m$ then set $q=n'$ and stop.

RTSP4. Reduce the cities as described in section 6.1.2.2.

RTSP5. Reduce the cost matrix as described in section 6.1.2.4.

RTSP6. Goto RTSP2.

Each of the original n cities can thus be grouped according to which node has replaced them. The optimum tour is found for each region (by exhaustive search, branch & bound, etc.) and the resultant subtours combined to approximate the optimum tour. It is found to be preferable to use a tour construction algorithm in the Euclidean case rather than the patching method used in the non-Euclidean case.

C1. Construct the optimal tour on each of the regions q. Define the 'distance' between each pair of regions to be the minimum cost of linking a city in one to a city in the other.

C2. Construct the MST for the regions using these distances.

C3. Construct a closed walk (a tour with repeated edges) over all cities which includes every subtour and passes from one region to another by using each MST edge twice.

C4. Transform the closed walk into a tour (Karp & Steele [1985]).

In this way, an approximation to the optimum tour is obtained. Results suggest that improvements of up to 10% are possible using this method over either the assignment problem approach or the arbitrary grouping method.

The ETSP is a well established and widely applicable problem. It is for this reason that it has been used here. In point of fact, the reduction technique will, on average, provide an improvement over the assignment problem solution for any case in which $c_{ij}=c_{ji}$ for all cities i and j (the symmetric TSP).

8.2. Initial Solutions, Perturbation and Lookahead Techniques

The patching algorithm presented in section 8.1.2.1 is not a true perturbation method since the initial solution presented to the algorithm is not a valid solution. Also it can be calculated precisely how many steps will be taken. It is, however, a perfectly natural extension of the ideas involved. The similarity between perturbation and lookahead methods has already been commented upon and this new concept allows a new, more general description of a <u>modification method</u> to be given. Perturbation methods, lookahead methods and refinement methods such as the patching algorithm are all special examples of the general case.

A modification method then consists of three components.

1. An initial solution I.
2. A final solution F.
3. An algorithm A which transforms I into F via a sequence of intermediate solutions S_0 (=I), S_1, S_2, ..., S_f (=F).

Clearly F must be a valid solution but S_0, S_1, S_2, ..., S_{f-1} need not (as is the case for the patching algorithm). The field of application of such a general form of method is extremely wide - the numerical solution of algebraic equations, vehicle routeing problems and linear programming methods for example.

8.2.1. Applications of Step by Step Modification Methods

This section briefly outlines two practical applications of variations of the general modification method. They correspond, in essence, to a perturbation and a lookahead technique.

8.2.1.1. Tree Perturbation Techniques

The design of offshore natural gas pipeline systems is considered by Rothfarb et al. [1970]. This paper uses an ingenious method of simplification in order to produce a solution tree. Gas flows along the links and branches towards a single node of the tree positioned on the shore. The determination of this tree is not described here since it relies heavily upon the particular pressure requirements of gas networks. It is the next stage which is of interest.

The new process starts with the initial tree T_1. By definition, this has no circuits and adding any extra link will produce exactly one circuit. The following algorithm attempts to generate improved trees.

```
TP1. Let the current tree $T_c$ be the initial tree $T_1$.

TP2. Let the temporary tree $T_t$ be the current tree $T_c$.

TP3. For each node i in $T_t$ do the following.

     TP3.1. For each node j, non-adjacent to i in $T_t$, do the
            following.

            TP3.1.1. Connect i and j in $T_t$. This will form a
                     circuit with links $(i,j)$, $(j,a_1)$, $(a_1,a_2)$,
                     ..., $(a_x,i)$.

            TP3.1.2. Remove each of the links $(j,a_1)$, $(a_1,a_2)$,
                     ..., $(a_x,i)$ in turn in $T_t$. If any give an
                     improvement in cost over $T_c$ then set $T_c$:=$T_t$
                     and goto TP2.

TP4. Let the final tree $T_f$ be the current tree $T_c$.
```

The addition of each link forms a circuit and the removal of another link in the same circuit produces a different tree.

This is the limit of the proposals by Rothfarb et al. [1970]. In the light of the previous discussion, however, the process can clearly be extended. A number of links, l_1, l_2, ..., l_x, could be added at each stage, causing the formation of y $(y \geq x)$ circuits. all appropriate combinations of z $(y \geq z \geq x)$ links can then be tested for removal to produce a number of different trees. The level of perturbation is thus improved at the expense of longer processing time.

8.2.1.2. Lookahead Link Swapping Methods

The link swapping method of Lin & Kernighan [1973] is regarded as an effective method of generating near optimal TSP tours. The technique begins with any (valid) initial tour and applies a sequence if link swaps, gradually transforming the current tour in the direction of decreasing cost. Again the precise details are not considered here. Despite the method being widely accepted, however, one weakness is clearly acknowledged.

The process has a non-trivial level of perturbation in the sense that a variable number of links can be swapped at any one time. It does, however, only consider one such set of links at a time. In other words, there is no lookahead. An important link could be removed at a certain stage which would then be replaced at some subsequent stage, holding up the process unnecessarily.

These problems can be partly overcome by using a simple lookahead method in which a series of sets of link swaps are considered. The time saved by avoiding erroneous removals has to be balanced against the extra time taken by the lookahead method. Lin & Kernighan [1973] suggest that a two stage process is most suitable.

8.2.1.2.1. A Combined Algorithm

The lookahead method of Lin & Kernighan [1973] suggests a combined algorithm with the reduction method. In essence, this could be as follows.

C1. Apply the reduction algorithm to the n cities to derive q groupings

C2. Find the optimal tour for each of the q groups.

C3. Combine these q subtour into a tour as described in section 8.1.2.2.

C4. Apply the Lin & Kernighan method to search for improvements.

This algorithm has a high chance of optimality due to the nature of the lookahead link swapping method combined with fast convergence provided by the accuracy of the initial solution.

CHAPTER 9

CONCLUSIONS

No! No! Sentence first - verdict afterwards
Lewis Carroll
Alice's Adventures in Wonderland, 1865

There are an enormous number of ways in which a given set of nodes (PABXs etc.) may be interconnected to provide for their particular communication requirements. It is completely impractical to generate and examine every possible solution in turn and the problem is such that no helpful structure exists allowing more efficient methods to be used. Link and switch costs are generally complex functions of traffic and distance and can only be determined when all details about the rest of the network are known. Often practical constraints must be placed upon the solution network which, in turn, have to be considered by any optimisation process. Heuristic methods of some description must be used.

One approach is to simplify the real problem in such a way that some existing optimisation technique may be used. If constant costs for links and switches are assumed then the NOP may be expressed and solved as linear programming problem. However this is hardly realistic. An alternative is to discount certain types of solution on the basis of some knowledge of the network. There may be some justification to this idea but there will always be exceptions for which such an approach behaves badly. Some of the techniques, however, which are designed for simplistic optimisation on the complete network may be used on smaller problems, such as the multidrop methods and the concentrator location algorithms.

Practical methods of optimisation which do not make particularly blatant assumptions or simplifications are mostly of the type which attempt to construct a solution rather than select one. The double drop method is such an approach. The accuracy of these methods can not be guaranteed and they may often still take a considerable time to run.

The approach suggested throughout this work is one in which the complete NOP is divided into a number of sections and solved sequentially in this form. The reduction algorithm presented will allow the choice of tandems to be made from a smaller set of nodes and, in turn, will decrease the complexity of this part of the process. The number of tandem sets tested can be further reduced by noting the convexity of the cost curve and the uniqueness of the optimum number of tandems. In addition, if a certain tandem network connection topology is temporarily fixed, the problem can be made even easier.

The set of tandems obtained by fixing the tandem network topology can not be guaranteed to be correct due to the assumption that is made in its calculation. A series of local transformations, or perturbations, can therefore be applied to the tandem locations and connection topology in order to search for improvements. When none can be found, the tandems are fixed permanently. The final interconnection network and traffic routeing strategy between these tandems can then be determined using a lookahead sequence of link removals. The best node to tandem links can be determined separately.

A number of practical examples have been tested. For moderate sized networks, the compound method for solving the NOP has been shown empirically to produce excellent results. Known exact results are impossible for large networks but since results suggest the reduction process to be accurate, it is expected that situation should not worsen.

The methods of reduction, perturbation and lookahead are extremely general and can be applied to a variety of different areas of optimisation. The method suggested for the ETSP produces superior results to comparable techniques.

The sponsoring establishment are extremely pleased with results.

APPENDIX A

BASIC ALGORITHMS AND TECHNIQUES

Some of the algorithms and methods suggested in the main body of the thesis require procedures and techniques which are not of direct relevance to the problem which the algorithm solves. They clearly must be available, however, for practical implementation. These procedures are presented here.

Unlike the majority of the algorithms in the main text, the following procedures are written in strict Pascal.

A.1. A Procedure to Generate Tandem Sets

The following procedure will generate, in turn, all combinations of M tandems from n nodes. It operates by reading in the current combination of tandems and calculating the next combination in a set sequence.

The set of nodes is labelled i=1,2,...,n. A set of M tandems is represented by T[1],T[2],...,T[M] where each T[j] j=1,2,..,M is chosen uniquely from the set of nodes. The starting combination must be given by T[j]=j j=1,2,..,M. The procedure will then generate all intermediate combinations up to T[j]=j+n-M j=1,2,..,M, after which the BOOLEAN variable, alldone, becomes TRUE.

```
PROCEDURE reselect(VAR alldone:BOOLEAN;VAR T:ARRAY[1..M] OF 1..n)
VAR       i,k          : INTEGER ;
          reselected : BOOLEAN ;
BEGIN
k := M+1 ; reselected := FALSE ; alldone := FALSE ;
WHILE NOT reselected DO
          BEGIN
          k := k-1 ;
          IF k=0 THEN
             BEGIN
```

```
                  alldone := TRUE ; reselected := TRUE ;
                  END
               ELSE IF T[k]<n-M+k THEN
                  BEGIN
                  reselected := TRUE ; T[k] := T[k]+1 ;
                  FOR i := k+1 TO M DO T[i] := T[k]+i-k ;
                  END ;
               END ;
     END ;
```

A.2. A Procedure to Generate Tandem Network Connection Topologies

The following procedure will generate, in turn, all possible tandem network connection topologies (including disconnected ones - these can be detected elsewhere within the calling program). It operates by reading in the current topology and calculating the next according to a fixed sequence.

The tandem network connection topology is represented by an M x M BOOLEAN matrix $C=(c_{ij})$ where c_{ij} is TRUE if T[i] is connected to T[j] and FALSE otherwise. The starting topology is C=<u>FALSE</u> and the final topology is c_{ij}=TRUE i<>j, c_{ii}=FALSE, i,j=1,2,..,M (where a tandem is taken as not connected to itself by convention). At this stage, the BOOLEAN variable, alldone, becomes TRUE.

```
     PROCEDURE reconnect(VAR c:ARRAY[1..M,1..M] OF BOOLEAN;
                         VAR alldone:BOOLEAN) ;
     VAR       i,j,i1,j1    : INTEGER ;
               reconnected : BOOLEAN ;
     BEGIN
     alldone := TRUE ;
     FOR i := 1 TO M-1 DO
         FOR j := i+1 TO M DO
             IF NOT c[i,j] THEN alldone := FALSE ;
     IF NOT alldone THEN
            BEGIN
            reconnected := FALSE ; i := M-1 ; j := M ;
     WHILE NOT reconnected DO
         BEGIN
         IF NOT c[i,j] THEN
              BEGIN
              reconnected := TRUE ;
              c[i,j] :: TRUE ; c[i,j] :: TRUE ;
              FOR i1 := 1 TO M-1 DO
                 FOR j1 := i1+1 TO M DO
                    IF (i1<>i) OR (j1>j) THEN
                      BEGIN
                      c[i1,j1] := FALSE ;
                      c[j1,i1] := FALSE ;
                      END ;
              END
         ELSE
              BEGIN
              IF j=i+1 THEN
                BEGIN
                i := i-1 ; j := M ;
                END
              ELSE j := j-1 ;
              END ;
         END ;
     END ;
END ;
```

A.3. Data Storage and Processing Techniques

Within the confines of a small computer, both processing and storage resources are at a premium. A typical micro-computer, such as the IBM PC XT or AT, has 8/16 bit arithmetic operation, less than 1Mbit of RAM and, probably, a 10/20Mbit fixed disk. The trade-off between software speed and size should be constantly in mind. Ultimately, the storage constraints will take priority and a piece of code will need to run for as long as it takes in order to fit it on the machine. This section briefly discusses some simple techniques for conserving RAM, fixed disk space and processing time (in that order of priority).

A.3.1. Reducing Array Sizes

Consider, as an example, the 2-D array d[1..n,1..n] OF REAL representing the distance between each pair of nodes in the network. It may be required to find the shortest (or longest) distance. This search will, in principle, invlove some coding of the form

```
FOR i := 1 TO n DO
    FOR j :+ 1 TO n DO
        ..........................
```

However, d[i,j]=d[j,i] and d[i,i]=0 for all i,j=1,2,..,n so this method posseses redundancy in its calculations. The process can be made more efficient by replacing this code by

```
FOR i := 1 TO n-1 DO
    FOR j := i+1 TO n DO
        ..........................
```

In general, for a <u>symmetrical</u> Q-D array, Q[1..n,1..n, .. ,1..n], scanning is most efficient as

```
FOR a := 1 TO n-Q+1 DO
    FOR b := a+1 TO n-Q+2 DO
        .
            .
                FOR p := o+1 TO n-1 DO
                    FOR q := p+1 TO n DO
                        ..........................
```

In this way, both storage and processing time are minimised.

A. 3. 2. Storing Data on the Fixed Disk.

Extremely large, multidimensional arrays are unwelcome in efficient pieces of code. Such data should be stored externally from the relevant program (i.e. on the fixed disk, etc.) and read in and written to as required.

This presents slight difficulties in positioning and finding elements of data. No multidimensional structure exists on external storage, only a sequential array of locations. It is possible to translate these to a number of files, each in the form of a list starting from position 0 and continuing 1, 2, etc.. One piece of data can be stored at each location. Some method is required, however, of transforming the raw data into this form.

Firstly, consider the special instance of the distance array in the previous section where $i=1,2,..,n-1$, $j=i+1,i+2,..,n$. The location of the $(i,j)^{th}$ element in a sequential file, starting from 0, will be

$$\begin{aligned} L(i,j) &= (n-1) + (n-2) + (n-3) + .. + (n-i+1) + j - i - 1 \\ &= n(i-1) - (1+2+..+i-1) + j - i - 1 \\ &= n(i-1) - i(i-1)/2 + j - i - 1 \end{aligned}$$

To convert from an integer location L back into (i,j) format, the following piece of code is required.

```
PROCEDURE convert(L:INTEGER;VAR i,j:INTEGER) ;
VAR K : INTEGER ;
BEGIN
i := 0 ;
K := L ;
WHILE K>0 DO
      BEGIN
      i := i+1 ;
      K := K-N+i ;
      END ;
j := L+i-n(i-1)+i(i-1)/2 ;
END ;
```

For the general array type with fully varying parameters, a simpler form is available. For a Q-D array Q[1..n,1..n, .. ,1..n], the sequential location of the $(a,b,..,q)^{th}$ element is given by

$$L(a,b,..,q) = n^{q-1}(a-1) + n^{q-2}(b-1) + .. + n(p-1) + (q-1)$$

For example, $L(i,j) = n(i-1) + (j-1)$ and $L(i,j,k) = n^2(i-1) + n(j-1) + (k-1)$. A similar, but simpler, routine to the above can be written for the inverse operation.

A.4. A Procedure to Generate Link Removal Vectors.

This procedure is used to systematically generate vectors of length 3r containing all possible arrangements of Ω tandems. It used in particular in the lookahead link removal process of section 6.4 where, at stage r', the link between tA[r'] and tB[r'] is removed and the traffic rerouted via tC[r'].

If the inital vector is (1,1,1,...,1) this procedure will generate each intermediate vector, in turn, up to (Ω,Ω,Ω,...,Ω). When this happens, the BOOLEAN variable alldone becomes true.

The final, optional, section of this serves to remove trivial vectors such as those in which the link between X and X is removed.

```
PROCEDURE reselect_vector(VAR tA,tB,tC:array[1..r] of 1..Ω;
                          VAR alldone:BOOLEAN) ;
VAR i,j                   : INTEGER ;
    acceptable,reselected : BOOLEAN ;
BEGIN
alldone :: TRUE ;
acceptable :: FALSE ;
FOR i :: 1 TO r DO IF (tA[i]<Ω) or (tB[i]<Ω) or (tC[i]<Ω)
                   THEN alldone :: FALSE ;
IF NOT alldone THEN
   WHILE NOT acceptable DO
         BEGIN
         j :: r+1 ;
```

```
reselected := FALSE ;
acceptable := TRUE ;
WHILE NOT reselected DO
    BEGIN
    j := j-1 ;
    IF tC[j]<Ω THEN
      BEGIN
      reselected := TRUE ;
      tC[j] := tC[j]+1 ;
      FOR i := j+1 TO r DO
          BEGIN
          tA[i] := 1 ;
          tB[i] := 1 ;
          tC[i] := 1 ;
          END ;
      END
    ELSE IF tB[j]<Ω THEN
          BEGIN
          reselected := TRUE ;
          tB[j] := tB[j]+1 ;
          tC[j] := 1 ;
          FOR i := j+1 TO r DO
              BEGIN
              tA[i] := 1 ;
              tB[i] := 1 ;
              tC[i] := 1 ;
              END ;
          END
    ELSE IF tA[j]<Ω THEN
          BEGIN
          reselected := TRUE ;
          tA[j] := tA[j]+1 ;
          tB[j] := 1 ;
          tC[j] := 1 ;
          FOR i := j+1 TO r DO
              BEGIN
```

```
                          tA[i] := 1 ;
                          tB[i] := 1 ;
                          tC[i] := 1 ;
                          END ;
                      END
                 ELSE IF j=1 THEN reselected := TRUE ;
                 END ;
          alldone := TRUE ;
          FOR i := 1 TO r DO
              BEGIN
              IF (tA[i]>=tB[i]) or (tA[i]=tC[i]) or (tB[i]=tC[i])
                 THEN acceptable := FALSE ;
              IF (tA[i]<Ω) or (tB[i]<Ω) or (tC[i]<Ω)
                 THEN alldone := FALSE ;
              END ;
          IF alldone THEN acceptable := TRUE ;
          END ;
END ;
```

APPENDIX B

ERLANG B TRAFFIC TABLES

The Grade of Service (GOS) of a transmission link is the probability that a single call will fail to transmit due to all available circuits being in use. This probability is given by

$$P(A,c) = \frac{A^c/c!}{\sum_{x=0}^{c} A^x/x!}$$

where A is the offered traffic in erlangs. and c is the number of available circuits. This is the Erlang equation of the first kind (Hills [1979]). Its derivation is based on a number of assumptions dealt with in chapter 4. As it is impossible to rearrange this into the form c = f(A,P), the information is usually tabulated by calculation. The following tables are those used for all calculations and tests described in the main text. They give the amount of traffic (in erlangs) which can be carried by the number of circuits in each row at the GOS in each column. A reverse lookup process is used to obtain the number of circuits required to carry a certain amount of traffic at a particular GOS.

GOS	0.001	0.005	0.01	0.02	0.03	0.05	0.1
Circuits							
1	0.001	0.005	0.01	0.02	0.03	0.05	0.11
2	0.05	0.11	0.15	0.22	0.28	0.38	0.60
3	0.19	0.35	0.46	0.60	0.72	0.90	1.27
4	0.44	0.70	0.87	1.09	1.26	1.52	2.05
5	0.76	1.13	1.36	1.66	1.88	2.22	2.88

6	1.15	1.62	1.91	2.28	2.54	2.96	3.76
7	1.58	2.16	2.50	2.94	3.25	3.74	4.67
8	2.05	2.73	3.13	3.63	3.99	4.54	5.60
9	2.56	3.33	3.78	4.34	4.75	5.37	6.55
10	3.09	3.96	4.46	5.08	5.53	6.22	7.51
11	3.65	4.61	5.16	5.84	6.33	7.08	8.49
12	4.23	5.28	5.88	6.62	7.14	7.95	9.47
13	4.83	5.96	6.61	7.41	7.97	8.83	10.47
14	5.45	6.66	7.35	8.20	8.80	9.73	11.47
15	6.08	7.38	8.11	9.01	9.65	10.63	12.48
16	6.72	8.10	8.87	9.83	10.51	11.54	13.50
17	7.38	8.83	9.65	10.66	11.37	12.46	14.52
18	8.05	9.58	10.44	11.49	12.24	13.38	15.55
19	8.72	10.33	11.23	12.33	13.11	14.31	16.58
20	9.41	11.09	12.03	13.18	14.00	15.25	17.61
21	10.11	11.86	12.84	14.04	14.89	16.19	18.65
22	10.81	12.64	13.65	14.90	15.78	17.13	19.69
23	11.52	13.42	14.47	15.76	16.68	18.08	20.74
24	12.24	14.20	15.29	16.63	17.58	19.03	21.78
25	12.97	15.00	16.12	17.50	18.48	19.99	22.83
26	13.70	15.80	16.96	18.38	19.39	20.94	23.88
27	14.44	16.60	17.80	19.26	20.30	21.90	24.94
28	15.18	17.41	18.64	20.15	21.22	22.87	26.00
29	15.93	18.22	19.49	21.04	22.14	23.83	27.05
30	16.68	19.04	20.34	21.93	23.06	24.80	28.11
31	17.44	19.86	21.19	22.83	23.99	25.77	29.17
32	18.20	20.68	22.05	23.73	24.91	26.75	30.23
33	18.97	21.51	22.91	24.63	25.84	27.72	31.30
34	19.74	22.34	23.77	25.53	26.77	28.70	32.36
35	20.52	23.17	24.64	26.43	27.71	29.68	33.43

36	21.30	24.01	25.51	27.34	28.64	30.66	34.50
37	22.08	24.85	26.38	28.25	29.58	31.64	35.57
38	22.86	25.69	27.25	29.17	30.51	32.63	36.64
39	23.65	26.54	28.13	30.08	31.45	33.61	37.71
40	24.44	27.38	29.01	31.00	32.39	34.60	38.79
41	25.24	28.23	29.89	31.92	33.34	35.59	39.86
42	26.04	29.08	30.77	32.84	34.29	36.58	40.94
43	26.84	29.94	31.66	33.76	35.24	37.57	42.01
44	27.64	30.80	32.54	34.68	36.19	38.56	43.09
45	28.45	31.66	33.43	35.61	37.15	39.55	44.16
46	29.26	32.52	34.32	36.53	38.10	40.54	45.24
47	30.07	33.38	35.21	37.46	39.06	41.54	46.32
48	30.88	34.25	36.11	38.39	40.02	42.54	47.40
49	31.69	35.11	37.00	39.32	40.98	43.54	48.48
50	32.51	35.98	37.90	40.25	41.93	44.53	49.56
51	33.3	36.9	38.8	41.2	42.9	45.5	50.6
52	34.2	37.7	39.7	42.1	43.9	46.5	51.7
53	35.0	38.6	40.6	43.1	44.8	47.5	52.8
54	35.8	39.5	41.5	44.0	45.8	48.5	53.9
55	36.6	40.4	42.4	44.9	46.7	49.5	55.0
56	37.5	41.2	43.3	45.9	47.7	50.5	56.1
57	38.3	42.1	44.2	46.8	48.7	51.5	57.1
58	39.1	43.0	45.1	47.8	49.6	52.6	58.2
59	40.0	43.9	46.0	48.7	50.6	53.6	59.3
60	40.8	44.8	46.9	49.6	51.6	54.6	60.4
61	41.6	45.6	47.9	50.6	52.5	55.6	61.5
62	42.5	46.5	48.8	51.5	53.5	56.6	62.6
63	43.3	47.4	49.7	52.5	54.5	57.6	63.7
64	44.2	48.3	50.6	53.4	55.4	58.6	64.8
65	45.0	49.2	51.5	54.4	56.4	59.6	65.8

66	45.8	50.1	52.4	55.3	57.4	60.6	66.9
67	46.7	51.0	53.4	56.3	58.4	61.6	68.0
68	47.5	51.9	54.3	57.2	59.3	62.6	69.1
69	48.4	52.8	55.2	58.2	60.3	63.7	70.2
70	49.2	53.7	56.1	59.1	61.3	64.7	71.3
71	50.1	54.6	57.0	60.1	62.3	65.7	72.4
72	50.9	55.5	58.0	61.0	63.2	66.7	73.5
73	51.8	56.4	58.9	62.0	64.2	67.7	74.6
74	52.7	57.3	59.8	62.9	65.2	68.7	75.6
75	53.5	58.2	60.7	63.9	66.2	69.7	76.7
76	54.4	59.1	61.7	64.9	67.2	70.8	77.8
77	55.2	60.0	62.6	65.8	68.1	71.8	78.9
78	56.1	60.9	63.5	66.8	69.1	72.8	80.0
79	57.0	61.8	64.4	67.7	70.1	73.8	81.1
80	57.8	62.7	65.4	68.7	71.1	74.8	82.2
81	58.7	63.6	66.3	69.6	72.1	75.8	83.3
82	59.5	64.5	67.2	70.6	73.0	76.9	84.4
83	60.4	65.4	68.2	71.6	74.0	77.9	85.5
84	61.3	66.3	69.1	72.5	75.0	78.9	86.6
85	62.1	67.2	70.0	73.5	76.0	79.9	87.7
86	63.0	68.1	70.9	74.5	77.0	80.9	88.8
87	63.9	69.0	71.9	75.4	78.0	82.0	89.9
88	64.7	69.9	72.8	76.4	78.9	83.0	91.0
89	65.6	70.9	73.7	77.3	79.0	84.0	92.1
90	66.5	71.8	74.7	78.3	80.0	85.0	93.1
91	67.4	72.7	75.6	79.3	81.0	86.0	94.2
92	68.2	73.6	76.6	80.2	82.0	87.1	95.3
93	69.1	74.5	77.5	81.2	83.0	88.1	96.4
94	70.0	75.4	78.4	82.2	84.0	89.1	97.5
95	70.9	76.3	79.4	83.1	85.0	90.1	98.6

96	71.7	77.2	80.3	84.1	86.8	91.1	99.7
97	72.6	78.2	81.2	85.1	87.8	92.2	100.8
98	73.5	79.1	82.2	86.0	88.8	93.2	101.9
99	74.4	80.0	83.1	87.0	89.8	94.2	103.0
100	75.2	80.9	84.1	88.0	90.8	95.2	104.1
101	76.1	81.8	85.0	88.9	91.8	96.3	105.2
102	77.0	82.8	85.9	89.9	92.8	97.3	106.3
103	77.9	83.7	86.9	90.9	93.8	98.3	107.4
104	78.8	84.6	87.8	91.9	94.8	99.3	108.5
105	79.7	85.5	88.8	92.8	95.7	100.4	109.6
106	80.5	86.4	89.7	93.8	96.7	101.4	110.7
107	81.4	87.4	90.7	94.8	97.7	102.4	111.8
108	82.3	88.3	91.6	95.7	98.7	103.4	112.9
109	83.2	89.2	92.5	96.7	99.7	104.5	114.0
110	84.1	90.1	93.5	97.7	100.7	105.5	115.1
111	85.0	91.1	94.4	98.7	101.7	106.5	116.2
112	85.9	92.0	95.4	99.6	102.7	107.5	117.3
113	86.7	92.9	96.3	100.6	103.7	108.6	118.4
114	87.6	93.8	97.3	101.6	104.7	109.6	119.5
115	88.5	94.7	98.2	102.5	105.7	110.6	120.6
116	89.4	95.7	99.2	103.5	106.7	111.7	121.7
117	90.3	96.6	100.1	104.5	107.7	112.7	122.8
118	91.2	97.5	101.1	105.5	108.7	113.7	123.9
119	92.1	98.4	102.0	106.4	109.7	114.7	125.0
120	93.0	99.4	103.0	107.4	110.7	115.8	126.1
121	93.8	100.3	103.9	108.4	111.6	116.8	127.2
122	94.7	101.2	104.9	109.4	112.6	117.8	128.3
123	95.6	102.2	105.8	110.3	113.6	118.9	129.4
124	96.5	103.1	106.8	111.3	114.6	119.9	130.5
125	97.4	104.0	107.7	112.3	115.6	120.9	131.6

126	98.3	105.0	108.7	113.3	116.6	122.0	132.7
127	99.2	105.9	109.6	114.2	117.6	123.0	133.8
128	100.1	106.8	110.6	115.2	118.6	124.0	134.9
129	101.0	107.8	111.5	116.2	119.6	125.1	136.0
130	101.9	108.7	112.5	117.2	120.6	126.1	137.1
131	102.8	109.6	113.4	118.2	121.6	127.1	138.2
132	103.7	110.6	114.4	119.1	122.6	128.2	139.3
133	104.6	111.5	115.3	120.1	123.6	129.2	140.4
134	105.5	112.4	116.3	121.1	124.6	130.2	141.5
135	106.4	113.4	117.2	122.1	125.6	131.3	142.6
136	107.3	114.3	118.2	123.1	126.6	132.3	143.7
137	108.2	115.2	119.1	124.0	127.6	133.3	144.8
138	109.1	116.2	120.1	125.0	128.6	134.3	145.9
139	110.0	117.1	121.0	126.0	129.6	135.4	147.0
140	110.9	118.0	122.0	127.0	130.6	136.4	148.1
141	111.8	119.0	122.9	128.0	131.6	137.4	149.2
142	112.7	119.9	123.9	128.9	132.6	138.5	150.3
143	113.6	120.8	124.8	129.9	133.6	139.5	151.4
144	114.5	121.8	125.8	130.9	134.6	140.5	152.5
145	115.4	122.7	126.7	131.9	135.6	141.6	153.6
146	116.3	123.6	127.7	132.9	136.6	142.6	154.7
147	117.2	124.6	128.6	133.8	137.6	143.6	155.8
148	118.1	125.5	129.6	134.8	138.6	144.7	156.9
149	119.0	126.5	130.6	135.8	139.6	145.7	158.0
150	119.9	127.4	131.6	136.8	140.6	146.7	159.1
151	120.8	128.3	132.5	137.8	141.6	147.8	160.3
152	121.7	129.3	133.5	138.7	142.6	148.8	161.4
153	122.6	130.2	134.5	139.7	143.6	149.8	162.5
154	123.5	131.2	135.4	140.7	144.6	150.9	163.6
155	124.4	132.1	136.4	141.7	145.6	151.9	164.7

156	125.3	133.0	137.4	142.7	146.6	152.9	165.8
157	126.2	134.0	138.3	143.6	147.6	153.9	166.9
158	127.1	134.9	139.3	144.6	148.6	154.9	168.0
159	128.0	135.9	140.2	145.6	149.6	156.0	169.1
160	129.0	136.8	141.2	146.6	150.6	157.0	170.2
161	129.9	137.7	142.2	147.6	151.7	158.0	171.3
162	130.8	138.7	143.2	148.6	152.7	159.1	172.4
163	131.7	139.6	144.1	149.6	153.7	160.1	173.5
164	132.6	140.6	145.1	150.5	154.7	161.1	174.6
165	133.5	141.5	146.1	151.5	155.7	162.2	175.7
166	134.4	142.5	147.0	152.5	156.7	163.2	176.8
167	135.3	143.4	148.0	153.5	157.7	164.2	177.9
168	136.2	144.3	148.9	154.5	158.7	165.3	179.0
169	137.1	145.3	149.9	155.5	159.7	166.3	180.1
170	138.1	146.2	150.8	156.5	160.7	167.4	181.2
171	139.0	147.2	151.8	157.5	161.7	168.4	182.3
172	139.9	148.1	152.8	158.5	162.7	169.4	183.4
173	140.8	149.1	153.7	159.4	163.7	170.5	184.5
174	141.7	150.0	154.7	160.4	164.7	171.5	185.6
175	142.7	151.0	155.6	161.4	165.7	172.6	186.7
176	143.6	151.9	156.6	162.4	166.7	173.6	187.8
177	144.5	152.8	157.5	163.4	167.7	174.7	188.9
178	145.4	153.8	158.5	164.4	168.7	175.7	190.0
179	146.3	154.7	159.4	165.4	169.7	176.8	191.1
180	147.3	155.7	160.4	166.4	170.7	177.8	192.2
181	148.2	156.6	161.4	167.4	171.8	178.8	193.4
182	149.1	157.6	162.3	168.4	172.8	179.9	194.5
183	150.0	158.5	163.3	169.3	173.8	180.9	195.6
184	150.9	159.5	164.3	170.3	174.8	181.9	196.7
185	151.8	160.4	165.2	171.3	175.8	183.0	197.8

186	152.8	161.4	166.2	172.3	176.8	184.0	198.9
187	153.7	162.3	167.2	173.3	177.8	185.0	200.0
188	154.6	163.3	168.1	174.3	178.8	186.1	201.1
189	155.5	164.2	169.1	175.3	179.8	187.1	202.2
190	156.4	165.2	170.1	176.3	180.8	188.1	203.3
191	157.4	166.1	171.0	177.3	181.8	189.2	204.4
192	158.3	167.1	172.0	178.3	182.8	190.2	205.5
193	159.2	168.0	173.0	179.3	183.8	191.2	206.6
194	160.2	168.9	173.9	180.3	184.8	192.3	207.7
195	161.1	169.9	174.9	181.2	185.9	193.3	208.8
196	162.0	170.8	175.9	182.2	186.9	194.3	209.9
197	162.9	171.8	176.8	183.2	187.9	195.4	211.0
198	163.8	172.7	177.8	184.2	188.9	196.4	212.1
199	164.7	173.7	178.7	185.2	189.9	197.4	213.2
200	165.6	174.6	179.7	186.2	190.9	198.5	214.3

These values are taken from Freeman [1984] and have been corrected where necessary. They are based on full circuit availability.

For larger quantities of traffic, a linear approximation may be used with small relative errors. For example with a GOS of 0.001 the number of circuits required to carry E erlangs can be estimated as [E+35.4] for E>165.6.

To convert from the given entry in erlangs to hundred call seconds per hour (ccs h^{-1}), multiply by 36. For example 200 circuits can carry 7714.8 ccs h^{-1} at a GOS of 0.1

APPENDIX C

PLYMOUTH POLYTECHNIC NETWORK OPTIMISATION PACKAGE

V.M. Grout - January 1987

CONTENTS

C.1. USER GUIDE

FOREWORD

The Plymouth Polytechnic Network Optimisation Package (PPNOP) is a software package for the design and analysis of telecommunication networks to be used on the IBM PC personal computer.

The package was designed and developed during 1984-1986 by Vic Grout and Peter Sanders (Department of Communication Engineering, Faculty of Technology, Plymouth Polytechnic, Drake Circus, Plymouth, Devon, PL4 8AA, UK) under the guidance of Ian Upton, Paul Reynolds and David White (Tallis Consultancy, British Telecom, 151 Gower Street, London, WC1E 6BA, UK).

This user guide has been written to conform to the standards presented in 'Tallis Systems Development Procedures and Standards', S.Holloway, T.Arnold & K.Pretorius, Part 5-4 (User Guide) (Holloway et al. [1986]).

PREFACE

This guide is divided into three main sections plus appendices and an index, the information contained within each section being as follows.

Section 1.1 is an introduction covering telecommunication networks and network optimisation, operation of the package and hardware requirements and options.

Section 1.2 describes the steps required in starting the package and reaching the top level menu.

Section 1.3 provides the user with a complete account of the package in its operational state. Each menu option is described and explained individually together with the implications of each such choice.

The appendices (section 2) contains an explanation of possible errors and a glossary of terms.

Section 3 is the index.

C.1.1. INTRODUCTION

C.1.1.1. Scope of Introduction

This Introduction consists of 4 parts.

This section describes the purpose of the others.

Section 1.1.2 provides a simple introduction to the principles of telecommunication networks and the requirements of network optimisation

Section 1.1.3 presents, in condensed form the facilities offered by the Plymouth Polytechnic Network Optimisation Package (PPNOP).

Section 1.1.4 outlines the hardware requirements needed to start and operate PPNOP as well as the options available regarding the different environments of its use.

C.1.1.2. Introduction to Network Optimisation

A source and/or sink of telecommunications traffic is called a node. A node is defined exactly by specifying its X and Y coordinates relative to some 2-dimensional fixed frame of reference. In the UK this is the Ordinance Survey National Grid, ranging from 0-700km west-east and 0-1000km south-north. Other information, such as names or numbers, may also be associated with the node, but the coordinates alone matter in the problem of optimisation.

Each node will require to send and/or receive information to/from other nodes in the plane. This information is telecommunications traffic or simply traffic. There are 2 possible types.

Switched Traffic. This type of traffic is handled by switching equipment at intermediate points on its journey from source to sink.

Non-Switched Traffic. This type of traffic is not handled by switching equipment at intermediate points on its journey. Even if it does not take the most direct route, the route is continuous - no switching takes place.

Switched traffic is measured in erlangs where 1 erlang represents an average of 1 call for one hour. The Erlang is a dimensionless unit. Non-Switched traffic is measured simply by the number of circuits provided for its transmission. The traffic from i to j, a_{ij}, can be considered as an entry in a traffic matrix A. The two types of traffic thus give rise to two traffic matrices.

The combination of node locations and traffic matrices completely describe the input to a network optimisation problem. It is referred to as an initial network. The problem of optimisation is to determine the best way of interconnecting these nodes so that the traffic can flow as required.

There are a number of parameters to the process which affect different parts of the problem.

The period of time that the solution network is designed to be used for, usually measured in years.

The grade of service (GOS) provided by the network, measured as the fraction of calls which fail to transmit.

The degree of rerouting, measured as the maximum number of extra stages in a path between two nodes that are not connected directly.

Those links which, for whatever reason, must not be removed or/and have extra traffic rerouted onto them.

In addition, there are a number of fixed constraints regarding the final appearance of the solution network.

Two types of links are available; single analogue circuits capable of carrying small levels of traffic and digital circuits, equivalent to a number of analogue circuits and more suitable for larger amounts of traffic.

The solution network has two levels. An appropriate subset of the set of nodes in the initial network is selected to form what is known as the core network. These nodes are then called transits and an alternative name for the core network is transit network. These transits are connected together in a partial mesh arrangement. The existence of a particular intertransit link depends on the characteristics of the network under investigation. The complete core network, however, must be connected. The other nodes are connected by a single link to their nearest transit. The lower level comprising ordinary nodes and the links connecting them up to the transits is known as the service or supply network.

Digital links are always used for the core network whereas either analogue or digital links can be used for the supply network, depending on which proves to be the cheapest.

A network problem then consists of an input network consisting of nodes and traffic. Parameters such as grade of service etc. are added and the result is a solution of the form described in the above constraints. The process is known as optimisation. The Plymouth Polytechnic Network Optimisation Package (PPNOP) provides a method for effecting this optimisation.

C.1.1.3. System Overview

The Plymouth Polytechnic Network Optimisation Package (PPNOP) is a complete system allowing problems of the form described in section 1.1.2 to be set up, solved and their results studied. More precisely, the facilities offered are as follows

Details of networks (nodes and traffic) can be set up using graphics input routines.

Such networks can be optimised completely or in part.

Partial solutions can be entered by hand followed by the completed optimisation.

Initial, partially optimised or optimised networks can be saved and retrieved as required.

Comprehensive results in written or graphical form can be produced on the screen or printer.

A more detailed description of these facilities together with instructions for their use is contained in section 1.3

C.1.1.4. Environment Overview

The Plymouth Polytechnic Network Optimisation Package (PPNOP) has been designed to operate on an IBM PC-XT or AT personal computer with 512K RAM. A monitor with a display card is, of course, essential, a printer is optional and a fixed disk is to be preferred to twin floppies. These points are elaborated upon in the following notes.

A monitor is obviously required. PPNOP is prepared to work with a monochrome monitor and a Hercules Graphics Card. This specification can only be changed by replacing the Hercules Graphics Card device driver by a device driver suitable for the monitor and card in use. This allows colour, enhanced resolution or enhanced graphics adaptors to be used if required. See the program documentation (Grout [1987b]) for details about changing monitor and/or graphics card.

PPNOP will operate on IBM compatible machines provided the previous point is considered and memory requirements are satisfied.

A printer is optional and is required only for producing permanent copies of results. Any printer which can be used with the machine in question will be suitable. High resolution is preferable for printing maps.

PPNOP operates at its fastest and in its most convenient state when used from a fixed disk. All of the software can be stored in the same place so there need be no changes of floppy discs. However, if required, the package can be run on a dual floppy machine with the necessary changes of discs between tasks. For larger networks, however, some rearrangement (or spreading out) of software may be necessary.

Permanent network data can be saved and retrieved on any drive at the request of the user. While the package is in operation, however, it is required that the current network (the one being worked on) be stored somewhere as well. The drive for storing the current network is set by default to drive C. If it is necessary to change this value (for working with two floppies, for example) it is necessary to consult the program documentation (Grout [1987b]). Otherwise, the location of the current data is of no interest to the user. For the remainder of this guide, it will be assumed that both operational software and current network data are stored on drive C. The description extends easily to other cases. With the hardware and software as described, section 1.2 explains how to get PPNOP up and running.

C.1.2. SYSTEM START-UP

The Plymouth Polytechnic Network Optimisation Package (PPNOP) is supplied on 3 floppy discs. It will be assumed in this description that a hard disc is to be used. If the software is to be used from the floppies, then backup copies should be made and used while the originals are stored away safely.

To set up the package for the first time proceed as follows. The use of a separate directory is entirely optional but can avoid confusion with other data stored on the fixed disk.

From the root directory of the fixed disk drive (at the C> prompt) type

```
C>MD PPNOP
```

This creates a directory for the package. Transfer control to this directory by typing

```
C>CD PPNOP
```

Put disc number 1 (input) in drive A. Type

```
C>COPY A:*.*
```

Repeat for discs 2 (output) and 3 (optimisation) and store all three discs safely away. The package is now resident on the fixed disk in the directory PPNOP. To start the package type

```
C>PLYMOUTH
```

Some information regarding graphical output is displayed followed by the PPNOP title screen. Eventually the top level menu appears. Section 1.3 explains, in detail, how to proceed with using the various facilities offered by the package.

Now that the package is installed it is unnecessary, in future, to repeat all of the above procedure.

Simply type

```
C>CD PPNOP
```

(if necessary) to transfer control to the correct directory and type

```
C>PLYMOUTH
```

to enter the package as before. Again, the top level menu is eventually displayed.

C.1.3. SYSTEM OPERATION

This section presents, in detail, the facilities available within the Plymouth Polytechnic Network Optimisation Package (PPNOP) and describes how to use them. The topics are divided into groups in the same way as they appear in the on-screen menus. For each section, a brief summary of that command group is given followed by a more detailed individual description.

C.1.3.1. Top Level Options

On entry to PPNOP, and at subsequent times after, the top level menu is displayed. This presents, in the most general form, the possible actions which may be taken. Entering Q or a digit from 1 to 7 selects the required subject or job area to be worked upon. Sections 1.3.2-9 describe each of these areas in detail with subsections covering individual jobs. In general, entering R at any stage will return to the previous stage or to the appropriate menu.

C.1.3.2. Set up Network Details

Entering 1 at the top level menu causes the Set up Network Details menu to be displayed. This collection of routines allows the user to define information for a new or changed network. In principle, it is the first step in any optimisation job to set up the network under consideration. Entering a digit from 1 to 7 will select a particular part of the network to be set up or modified. Entering R will cause the top level menu to be displayed again. Sections 1.3.2.1-7 describe each option within this menu in greater detail.

C.1.3.2.1. Full Network Node Details

Entering 1 at the Set up Network Details menu begins the process of setting up initial network nodes. A description of the routine is presented over two pages followed by an invitation to input textual information to accompany the network. If this is required, enter Y and enter the appropriate personal identification and network name. Otherwise enter N. The number of nodes is then requested and the correct integer should be entered. It is now time to enter individual details for each node.

A map of the UK is displayed and details of the first node are requested. The serial number is already provided and the name and coordinates are needed. Enter the name of the node (any string of characters). Enter the X-coordinate (a number from 0 to 700) and Y-coordinate (a number from 0 to 1000). The node will now flash at the appropriate place on the map as specified by the coordinates. Check to see if the input data is correct. If there is any error, enter N. The node will disappear and The information will be rerequested. Otherwise enter Y (or press [RETURN]) to continue with the other nodes in the network. The flashing node is replaced by a smaller permanent marker to avoid confusion and details of the second node are requested.

This process is repeated for each node in the network. When all node information has been entered in this way, press any key to return to the Set up Network Details menu.

C.1.3.2.2. Full Network Traffic Details

Entering 2 at the Set up Network Details menu begins the process of defining the traffic flow in the network. Firstly a page of textual information is displayed. It is then possible to either produce two new traffic matrices or modify two existing ones. Select 1 or 2 as required. The required source node (row of matrix) is then requested. Enter a number between 1 and the number of nodes in the network. This row can be entered (updated) sequentially or selectively. This means that the elements of the row can be worked through in order or the appropriate columns can be selected in turn. Enter A or B as required. If B is selected, the user is prompted for each destination node as required and once the traffic data is entered, the user is asked whether there are any more.

For each node pair (element in the matrix), two values are required to represent the switched and non-switched traffic. The non-switched traffic should be an integer. Non-integers are rounded down. Enter both amounts as required.

At the end of each row (sequentially) or following a N response to the 'Another destination node?' question (selectively), the user is asked whether another source node (matrix row) is to be entered (updated). If the response is Y, the process is repeated. If N is entered, the traffic set up procedure is finished and the Set up Network Details menu is redisplayed.

C.1.3.2.3. Transit Network Node Details

Entering 3 at the Set up Network Details menu begins the process of setting up transit nodes. In a similar fashion to the Set up Full Network Node Details option, a description of the routine is presented over two pages followed by an invitation to input textual information to accompany the network. If this is required, enter Y and enter the appropriate personal identification and network name. Otherwise enter N. The number of transits is then requested and the correct integer should be entered. It is now time to enter individual details for each transit.

A map of the UK is displayed and details of the first transit are requested. The serial letter is already provided and the name and coordinates are needed. Enter the name of the transit (any string of characters). Enter the X-coordinate (a number from 0 to 700) and Y-coordinate (a number from 0 to 1000). The transit will now flash at the appropriate place on the map as specified by the coordinates. Check to see if the input data is correct. If there is any error, enter N. The transit will disappear and The information will be rerequested. Otherwise enter Y (or press [RETURN]) to continue with the other transits in the network. The flashing transit is replaced by a smaller permanent marker to avoid confusion and details of the second transit are requested.

This process is repeated for each transit in the core network. When all transit information has been entered in this way, press any key to return to the Set up Network Details menu.

C.1.3.2.4. Transit Network Traffic Details

Entering 4 at the Set up Network Details menu begins the process of defining the traffic flowing within the core network. In a similar fashion to the Set up Full Network Traffic Details option, a description of the routine is displayed followed by the choice of creating new traffic details or modifying existing details. Select 1 or 2 as required. The part of the matrix to be written or updated is then requested. Enter the required start and finish elements. For example a start element of (2,6) and a finish element of (5,3) would cause all but the first five elements of row 2, all of rows 3 and 4 and the first 3 elements of row 5 to be written or updated. To enter (update) all elements of the matrices, therefore, enter 1, 1, M and M where M is the number of transits in the core network. For each transit pair, enter the switched and non-switched traffic in the same way as for nodes.

When all required transit pairs have been dealt with, the Set up Network Details menu is redisplayed.

C.1.3.2.5. Cost Details

Entering 5 at the Set up Network Details menu begins the process of defining the cost functions. The cost functions for analogue links, digital links and transits can be set by the user although preset functions are available if needed. Enter 1 or 2 as required.

There are three components to the cost function; analogue link costs, digital link costs and transit costs. The first to be dealt with are the analogue link costs. The cost functions may be step functions. In the case of analogue links the function may take a different form for various steps of distance. Enter the number of steps (1 if the function is not a step function). The step limits are then requested (if there are more than one step). Enter each upper limit as required. When all step limits have been entered, the actual costs for each step are requested. The facility exists to have a different cost for the first circuit an subsequent circuits. Within each of these divisions there are connection costs and annual rental charges. Within each of these subdivisions there are fixed costs and charges per unit distance. This makes 8 components to the analogue link costs. Enter each component as required. This process is then repeated for each step in the analogue cost function.

This entire process is now repeated for digital links with exactly the same system as for analogue links.

The step option also exists for transits costs. Here the step limits are the number of input/output ports required (in other words the number of circuits entering or leaving the transit). Enter the number of steps and the step limits as before. There are 4 components to the transit costs; installation cost and annual rental, each of which is subdivided into a fixed cost and a charge per port. Enter each of the 4 components in turn and repeat for each step.

When all costs have been entered the Set up Network Details menu is redisplayed.

C.1.3.2.6. Reduce Traffic Details

If the node and traffic details have been set up along with the transit details as described in sections 1.3.2.1-3 it is unnecessary to explicitly define the transit traffic details. This can be worked out by reducing the original traffic details in the correct form. Entering 6 at the Set up Network Details menu produces the transit traffic details automatically in this way. The Set up Network Details menu is then redisplayed.

C.1.3.2.7. Modify Existing Network Details

Entering 7 at the Set up Network Details menu begins the process of updating or modifying the existing full network details. Nodes can be added, deleted or modified as required.

Firstly the numbers of nodes to be added, deleted and modified are requested. If there are no nodes in any particular category then enter 0. The serial number of each node to be deleted is then requested. Enter each one in turn. These nodes are automatically removed from the initial network.

Next the serial number of each node to be changed is requested. Enter each one in turn. For each node enter the new details as appropriate. This update is made to the initial network at once.

Finally each of the nodes to be added must be named and given X and Y coordinates. Then the switched and non-switched traffic between that node and each of the existing nodes must be given in the form described in section 1.3.2.2.

When all additions, deletions and modifications have been carried out the Set up Network Details menu is redisplayed.

C.1.3.3. Retrieve Saved Network Details

Entering 2 at the top level menu selects the retrieve options. Any network, whole or complete, initial, partially optimised or optimised can be retrieved from any drive, as can the cost functions. The drive to be searched is the default data drive. This is initially set to drive C. Entering a digit from 1 to 4 will select the particular details to be retrieved, 5 will allow the default data drive to be changed and R will return to the top level menu.

C.1.3.3.1. Full Network Details

Entering 1 at the Retrieve Network Details menu retrieves an initial network (nodes, traffic and cost function) from the default drive. The name of the network must be entered. Once the network has been retrieved, the Retrieve Network Details menu is redisplayed.

C.1.3.3.2. Core Network Details

Entering 2 at the Retrieve Network Details menu retrieves a core network (transits, traffic and cost function) from the default drive. The name of the core network must be entered. Once the core network has been retrieved, the Retrieve Network Details menu is redisplayed.

C.1.3.3.3. Optimised Network Details

Entering 3 at the Retrieve Network Details menu retrieves an optimised network (nodes, transits, traffic, cost functions, circuits, grade of service, period of optimisation etc.) from the default drive. The name of the optimised network must be given. Once the optimised network has been retrieved, the Retrieve Network Details menu is redisplayed.

C.1.3.3.4. Cost Details

Entering 4 at the Retrieve Network Details menu retrieves a cost function from the default drive. The name under which the costs are stored must be given. Once the costs have been retrieved, the Retrieve Network Details menu is redisplayed.

C.1.3.3.5. Change Default Data Drive

Entering 5 at the Retrieve Network Details menu allows the default data drive to be changed. The user is asked for the new drive and then the Retrieve Network Details menu is redisplayed. Any future retrievals will be made from this new drive (until this option is used again).

C.1.3.4. Optimise Network

Entering 3 at the top level menu causes the Optimise Network menu to be displayed. This set of options form the major work area of the package. Entering a digit from 1 to 3 selects one of the optimisation jobs to be performed. Option 1 offers a complete optimisation process while 2 and 3 allow the problem to be analysed in stages. Entering R causes the top level menu to be redisplayed. Some of these procedures can take a considerable time to execute. If wished, [CNTRL-BREAK] can be used to terminate execution and return to the Optimise Network menu.

C.1.3.4.1. Optimise Full Network

Entering 1 at the Optimise Network menu selects the Optimise Full network option. This process provides a complete optimisation of an initial network. From the initial set of nodes, a set of transits is chosen and the interconnection of these transits is then determined.

Firstly the grade of service (GOS) is requested, followed by the period of optimisation. The GOS is a number between 0 and 1 (although only numbers between 0 and 1/3 are at all realistic - see section 1.1.2) and the period of optimisation is an integer number of years. Enter the appropriate values as required.

The initial network is now displayed. Press a key to start the optimisation process. The approximate details of the stage which the routine has reached are displayed at all times. These are of no great value to the user and are explained in the software documentation (Grout [1987b]). Eventually the current best set of transits is displayed and the user is asked if it is required that the system look further for a better solution. If the reply is positive the process of transit selection continues again otherwise it terminates.

Eventually a set of transits will be chosen. It now remains to optimise the connection strategy for the links between them. More information is required from the user in the way of problem parameters. The GOS for the core network is requested. This may or may not be the same as the GOS for the supply network. Then the number of years of operation of the core network is requested, followed by the degree of rerouting (the maximum number of extra transits in the path between two transits which are not directly connected) and the level of accuracy required. This ranges from 1 (low accuracy) to 5 (high accuracy). Enter the appropriate choices for each. Finally the number of fixed links (those which can or must not be removed) and the number of links which can or must not carry any extra traffic is requested. If there are any links in either category, then enter the appropriate quantity and instruct the system which links they are in response to the prompt.

The process of determining the best intertransit connection strategy is then ready to start. Press a key to begin. The display shows links being gradually removed until the solution network is obtained. Press a key to return to the Optimise network menu.

C.1.3.4.2. Determine Best Set of Transits

Entering 2 at the Optimise Network menu begins the process of selecting the best set of transits from a group of nodes. This process is, in fact, a single part of option 1 (section 1.3.4.1). The crucial difference is that the user is permitted to specify how many transits there must be instead of leaving it up to the package.

On entry to the process the user is prompted for the grade of service as before but is then asked for the number of transits required before the period of optimisation. See section 1.3.4.1 for full details.

C.1.3.4.3. Optimise Interconnection

Entering 3 at the Optimise Network menu selects the core network optimisation option. This process finds the best interconnection strategy for a number of transits already in existence. It is, in effect, the last stage of option 1 (section 1.3.4.1). The parameters such as grade of service and period of optimisation must be entered as described in section 1.3.4.1.

C.1.3.5. Save Network Details

Entering 4 at the top level menu selects the save options. Any network, whole or complete, initial, partially optimised or optimised can be saved on any drive, as can the cost functions. The drive to be used is the default data drive. This is initially set to drive C. Entering a digit from 1 to 4 will select the particular details to be saved, 5 will allow the default data drive to be changed and R will return to the top level menu.

C.1.3.5.1. Full Network Details

Entering 1 at the Save Network Details menu saves an initial network (nodes, traffic and cost function) on the default drive. The name of the network must be entered. Once the network has been saved, the Save Network Details menu is redisplayed.

C.1.3.5.2. Core Network Details

Entering 2 at the Save Network Details menu saves a core network (transits, traffic and cost function) on the default drive. The name of the core network must be entered. Once the core network has been saved, the Save Network Details menu is redisplayed.

C.1.3.5.3. Optimised Network Details

Entering 3 at the Save Network Details menu saves an optimised network (nodes, transits, traffic, cost functions, circuits, grade of service, period of optimisation etc.) on the default drive. The name of the optimised network must be given. Once the optimised network has been saved, the Save Network Details menu is redisplayed.

C.1.3.5.4. Cost Details

Entering 4 at the Save Network Details menu saves a cost function on the default drive. The name under which the costs are to be stored must be given. Once the costs have been saved, the Save Network Details menu is redisplayed.

C.1.3.5.5. Change Default Data Drive

Entering 5 at the Save Network Details menu allows the default data drive to be changed. The user is asked for the new drive and then the Save Network Details menu is redisplayed. Any future saves will be made on this new drive (until this option is used again).

C.1.3.6. Print Network Details and Results

Entering 5 at the top level causes the Print Network Details and Results menu to be displayed. This group of options is concerned with the output of information of various sorts to the user. Output can be directed to the screen or to a printer if one is connected to the system. Entering a digit from 1 to 7 selects a particular aspect of the network to be output. Sections 1.3.6.1-7 describe each option in detail. Entering R causes the top level menu to be redisplayed.

C.1.3.6.1. Full Network Details

Entering 1 at the Print Network Details and Results menu selects the initial network as being the output required. The choice is then offered of sending the output to the screen or printer. Enter 1 or 2 as required. Next it is possible to select which part of the information is needed; the node details, traffic details or both. Enter 1, 2 or 3 as required. The appropriate part of the initial network is then sent as output to the appropriate device.

If output is directed to the screen, one page at a time is printed followed by - MORE -. Press any key to continue with the next page. When output is completed the Print Network Details and Results menu is redisplayed.

C.1.3.6.2. Transit Network Details

Entering 2 at the Print Network Details and Results menu selects the core network as being the output required. The choice is then offered of sending the output to the screen or printer. Enter 1 or 2 as required. Next it is possible to select which part of the information is needed; the transit details, traffic details or both. Enter 1, 2 or 3 as required. The appropriate part of the core network is then sent as output to the appropriate device.

If output is directed to the screen, one page at a time is printed followed by - MORE -. Press any key to continue with the next page. When output is completed the Print Network Details and Results menu is redisplayed.

C.1.3.6.3. Cost Details

Entering 3 at the Print Network Details and Results menu selects the cost functions as being the output required. The choice is then offered of sending the output to the screen or printer. Enter 1 or 2 as required. The cost functions are then sent as output to the appropriate device.

If output is directed to the screen, one page at a time is printed followed by - MORE -. Press any key to continue with the next page. When output is completed the Print Network Details and Results menu is redisplayed.

C.1.3.6.4. Optimised Network Details

Entering 4 at the Print Network Details and Results menu selects the optimised network as being the output required. The choice is then offered of sending the output to the screen or printer. Enter 1 or 2 as required. The optimised network is then sent as output to the appropriate device.

If output is directed to the screen, one page at a time is printed followed by - MORE -. Press any key to continue with the next page. When output is completed the Print Network Details and Results menu is redisplayed.

C.1.3.6.5. Optimised Network Costs

Entering 5 at the Print Network Details and Results menu selects the optimised network costs as being the output required. The choice is then offered of sending the output to the screen or printer. Enter 1 or 2 as required. The optimised network costs are then sent as output to the appropriate device.

When output is completed the Print Network Details and Results menu is redisplayed.

C.1.3.6.6. Adjusted Network Details

It is possible, at this stage to attempt some improvements to the supply network. Until this point, only digital links have been used. Now analogue links can be tried if wished (see the documentation (Grout [1987b]) for a fuller description of the use of analogue and digital links throughout the course of the optimisation). Entering 6 at the Print Network Details and Results menu invokes this process.

This output takes the same form as that produced by options 4 and 5 (sections 1.3.6.4 and 1.3.6.5) with those links proving to be cheaper with analogue circuits adjusted accordingly. As before, output can be directed to the screen or printer.

If output is directed to the screen, one page at a time is printed followed by - MORE -. Press any key to continue with the next page. When output is completed the Print Network Details and Results menu is redisplayed.

C.1.3.6.7. Display Map of Network

Entering 7 at the Print Network Details and Results menu causes a map of the UK to be displayed on the screen with the optimised network superimposed upon it. Initially the whole map is shown. Selected areas can be viewed in close up by zooming in. To do this enter Y to the Zoom in? prompt. Next enter the X and Y coordinates of the centre of the required viewing window (0-700 west-east, 0-1000 south-north) and the scale of magnification needed (1 normal, 2 twice as big etc.). Finally it is possible to suppress display of each of the network components (nodes, transits, supply network links and core network links) if wished (to improve clarity of display for example). For each of the four categories answer Y to have that component displayed and N not to have that component displayed. The map is then redrawn in the new form.

This process can be repeated as often as required. In addition, entering R in response to the Zoom in? prompt will immediately display the complete map in the original form.

At any stage the screen dump facility (usually Shift-PrtSc but it depends on the graphics card) can be used to produce a copy of the map on the printer.

Typing N in response to the Zoom in? prompt causes the map to be removed and the Print Network Details and Results menu to be redisplayed.

C.1.3.7. List Networks on Default Data Drive

Entering 6 at the top level menu causes a list of all networks stored on the default data drive to be displayed. The list is given one page at a time. Press any key to see the next page. When the complete list has been shown, the top level menu is redisplayed.

C.1.3.8. Delete the Current Network

Entering 7 at the top level menu selects the Delete Network option. All details of the current network are made ready to be deleted (lost forever). The user is asked to confirm this action. If this is confirmed the network is duly removed. In either case the top level menu is redisplayed at the conclusion.

C.1.3.9. Leave Plymouth System

Entering Q at the top level menu prepares to leave PPNOP. The user is asked to confirm that this is required and that all appropriate network data has been saved. If both points are confirmed then PPNOP is quit. If either answer is negative the quit is aborted and the top level menu is redisplayed.

C.2. APPENDICES

C.2.1. ERRORS

There are two ways in which the user can interact with the Plymouth Polytechnic Network Optimisation Package (PPNOP). Firstly there is the menu section of the package and secondly there are the occasions within each routine when data is required as input.

In response to any menu the only acceptable input is one of the digits listed as options within that menu or, if appropriate, R (return), Q (quit), Y (yes) or N (no). Any other response is not accepted. The input is removed from the screen and the menu is redisplayed. There is thus no room for error.

Similarly if a legal response is entered which invokes an option which is inappropriate at that point (such as asking for optimised network output before the network has even been optimised or trying to retrieve data which is not on the specified default data drive) that request is also ignored and the menu is redisplayed. Under certain circumstances a DOS error message will be briefly visible before returning to the appropriate menu.

For the case of data input only appropriate values are accepted. Any illegal data is ignored and the input request is repeated.

C.2.2. GLOSSARY OF TERMS

Adjusted Network
- Optimised network in which a comparison has been made for each node-transit link to see whether analogue or digital circuits would be preferable.

Analogue Circuits
- Single circuits suitable for carrying small amounts of data.

Analogue Link Costs
- Cost incurred by analogue circuits. Part 1 of cost function.

Circuit
- Hardware used to carry traffic between nodes.

Core Network
- Network comprising transits and digital links connecting them.

Default Data Drive
- Drive on which all data will be stored and from which all data will be retrieved.

Degree of Rerouting (DOR)
- Maximum number of extra transits in the path between any two transits not connected directly.

Digital Circuit
- Large traffic carriers equivalent to a number of analogue circuits and suitable for larger amounts of traffic.

Digital Link Costs
- Cost incurred by digital circuits. Part 2 of cost function.

Erlang
- Unit of telecommunications traffic. 1 erlang = 1 call for 1 hour.

Grade of Service (GOS)

- Fraction of all generated calls which fail to transmit.

Initial Network

- Complete set of nodes and traffic matrices to be optimised.

Network Optimisation

- Process of determining best connection strategy for a set of nodes.

Node

- Source and/or sink of telecommunications traffic.

Non-Switched Traffic

- Traffic which is not handled by switching equipment at intermediate points on its journey.

Period of Optimisation

- Length of time (usually in years) for which the network is designed to operate.

Service Network

- Low level network comprising nodes and node-transit links.

Supply Network

- Low level network comprising nodes and node-transit links.

Switched Traffic

- Traffic handled by switching equipment at intermediate points on its journey.

Traffic

- Information flowing from a source to a sink.

Traffic Matrix

- 2-Dimensional array of which the element in row i and column j represents the amount of traffic flowing from source i to sink j.

Transit
- Node selected to collect and distribute traffic around the top level network.

Transit Switch Costs
- Cost incurred by the use of a transit. Part 3 of cost function.

Transit Network
- Network comprising transits and digital links connecting them.

C.3. INDEX

REFERENCES

A single word even may be a spark of inextinguishable thought

Percy Bysshe Shelley

A Defence of Poetry, 1821

Aho, A.V., Hopcroft, J.E. & Ullman, J.D., 'The Design and Analysis of Computer Algorithms', Addison-Wesley, 1974.

Anderberg, M.R., 'Cluster Analysis for Applications', Academic Press, 1973.

Anderson, I., 'A First Course in Combinatorial Mathematics', Oxford University Press, 1974.

Bahl, L.R. & Tang, D.T., 'Optimisation of Concentrator Locations in Teleprocessing Networks', Symposium on Computer-Communications Networks and Teletraffic, Polytechnic Institute of Brooklyn, April 4th-6th , 1972.

Balas , E. & Christofides, N., 'A Restricted Lagrangian Approach to the Traveling Salesman Problem', Mathematical Programming, Vol 21, 1981, pp19-36.

Balfour, A. & Marwick, D.H., 'Programming in Standard FORTRAN 77', Heinemann Educational Books, 1979.

Bear, D., 'Principles of Telecommunication-Traffic Engineering', Peter Peregrinus, 1976.

Beker, H. & Piper, F., 'Cipher Systems: The Protection of Communications', Northwood, 1982.

Bellman, R.E., 'Dynamic Programming Treatment of the Traveling

Salesman Problem', Journal of the Association for Computing Machinery', Vol 9, 1962, pp61-63.

Biggs, N.L., Lloyd, E.K. & Wilson, R.J., 'Graph Theory 1736-1936', Clarendon Press, 1976.

Boardman, K.D. & Fickling, K., 'A Private Integrated Digital Data Network for London Electricity', First IEE Conference of UK Telecommunications Networks: Present and Future, June 2-3 1987.

Bondy, J.A. & Murty, U.S.R., 'Graph Theory with Applications', Macmillan, 1976.

BP, 'Group (UK) Telecommunications Network Design Study: Main Report' (restricted), Report No. TEL 00184, February 1984.

Brady, J.M., 'The Theory of Computer Science: A Programming Approach', Chapman & Hall, 1977.

Brockmeyer, E., Halstrom, H.L. & Jensen, A., 'The Life and Works of A.K. Erlang', Copenhagen Telephone Company, 1948.

Brown, D.W. & Lidbetter, E.J., 'The Future Network', British Telecommunications Engineering, Vol 3, Jan 1985, pp318-321.

Cayley, A., 'A Theorem on Trees', Collected Papers, Cambridge, 1897, pp26-28.

Chandy, K.M. & Russell R.A., 'The Design of Multipoint Linkages in a Teleprocessing Tree Network', IEEE Transactions on Computers, Vol C-21, No.10, 1972, pp1062-1066.

Christofides, N., 'Worst Case Analysis of a New Heuristic for Travelling Salesman Problem', Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, 1976.

Christofides, N., Mingozzi, A., Toth, P. & Sandi, C., 'Combinatorial Optimisation', Wiley, 1979.

Cottam, G. & Curtis, J.T., 'A Method for Making Rapid Surveys of Woodlands by Means of Pairs of Randomly Selected Trees', Ecology, Vol 30, No.1, January 1949, pp101-104.

Cutland, N.J., 'Computability: An Introduction to Recursive Function Theory', Cambridge University Press, 1980.

Dantzig, G.B., 'Linear Programming and Extensions', Princetown University Press, 1963.

Dantzig, G.B., Fulkerson, D.R. & Johnson, S.M., 'On a Linear Programming Combinatorial Approach to the Traveling Salesman Problem', Operations Research, Vol 7, 1959, pp58-66.

Deo, N., 'Graph Theory with Applications to Engineering and Computer Science', Prentice Hall, 1974.

Derman, C., Gleser, L.J. & Olkin, I., 'A Guide to Probability Theory and Application', Holt, Rinehart & Winston, 1973.

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

Dirac, G.A., 'Some Theorems on Abstract Graphs', Proceedings of the London Mathematical Society, Series 3, Vol 2, 1952, pp69-81.

Edmonds, J., 'The Chinese Postman's Problem', Operations Research, Vol 13, Supplement 1, B73, 1965,

Edmonds, J. & Johnson, E.L., 'Matching, Euler Tours and the Chinese Postman', Mathematical Programming, Vol 5, 1973, pp88-124.

Elias, P., Feinstein, A. & Shannon, C.E., 'A Note on the Maximum Flow

Through a Network', IRE Transactions on Information Theory, Vol IT-2, 1956, pp117-119.

Elias, D. & Ferguson, M.J., 'Topological Design of Multipoint Teleprocessing Networks', IEEE Transactions on Communications, Vol COM-22, No.11, November 1974, pp1753-1762.

Esau, L.R. & Williams, K.C., 'A Method for Approximating the Optimal Network', IBM System Journal, Vol 5, No.3, 1966, pp142-147.

Findlay, W. & Watt, D.A., 'Pascal: An Introduction to Methodical Programming', Pitman, 1978.

Flood, J.E., 'Mathematical Models', Telecommunication Networks, J.E. Flood (ed.), Peter Peregrinus, 1975, pp300-317.

Ford, L.R. & Fulkerson, D.R., 'Maximal Flow Through a Network', Canadian, Journal of Mathematics, Vol 8, 1956, p399-404.

Frank, H. & Chou, W., 'Routing in Computer Networks', Networks, Vol 1, 1971, pp99-112.

Freeman, R.L., 'Reference Manual for Telecommunications Engineering', Wiley, 1984.

Fryer, M.J., 'An Introduction to Linear Programming and Matrix Game Theory', Arnold, 1978.

Garbutt, B.N., 'Digital Restructuring of the British Telecom Network', British Telecommunications Engineering, Vol 3, Jan 1985, pp300-303.

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

Gerald, C.F., 'Applied Numerical Analysis. Second Edition', Addison Wesley, 1978.

Golden, B.L., 'A Statistical Approach to the TSP', Networks, Vol 7, 1977, pp209-225.

Graham, R.L., 'An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set', Information Processing Letters, Vol 1, 1972, pp132-133.

Grout, V.M. (a), 'Plymouth Polytechnic Network Optimisation Package - User Guide', British Telecom Tallis Consultancy, 1987.

Grout, V.M. (b), 'Plymouth Polytechnic Network Optimisation Package - Software Documentation', British Telecom Tallis Consultancy, 1987.

Grout, V.M., Sanders, P.W. & Stockel, C.T. (a), 'Practical Approach to the Optimisation of Large Scale Switching Networks', Computer Communications, Vol 10, No.1, February 1987, pp30-38.

Grout, V.M., Sanders, P.W. & Stockel, C.T. (b), 'Reduction Techniques Providing Initial Groupings for Euclidean Travelling Salesman Patching Algorithms', Submitted to Applied Mathematical Modelling, November 1987.

Grout, V.M., Sanders, P.W. & Stockel, C.T., 'Large Scale Telecommunication Network Optimisation', Simulation and Optimization of Large Systems, A.J. Osiadacz (ed.), Oxford University Press, 1988, pp263-283.

Gurrie, M.L. & O'Connor, P.J., 'Voice/Data Telecommunications Systems', Prentice-Hall, 1986.

Hai-Hoc, H. (a), 'Topological Optimisation of Networks: A Nonlinear Mixed Integer Model Employing Generalised Benders Decomposition', IEEE Transactions on Automatic Control, Vol AC-24, No.1, February 1982, pp164-169.

Hai-Hoc, H. (b), 'Lagrangean Heuristics for Computer Computer

Communication Network Optimisation', Department of Electrical Engineering, Ecole Polytechnique, Montreal, Canada, 1982.

Harary, F., 'Graph Theory', Addison-Wesley, 1971.

Harary, F. & Palmer, E.F., 'Graphical Enumeration', Academic Press, 1973.

Heap, S. & Arthur, J.D., 'Network Management in the Digital Network', British Telecommunications Engineering, Vol 3, Jan 1985, pp308-310.

Held, M. & Karp, R.M., 'A Dynamic Programming Approach to Sequencing Problems', SIAM Journal of Applied Mathematics, Vol 10, 1962, pp196-210.

Hills, M.T., 'Telecommunication Switching Principles', Allen & Unwin, 1979.

Hoare, C.A.R. & Wirth, N., 'An Axiomatic Definition of the Programming Language PASCAL', Acta Informatica, Vol 2, 1972, pp335-355.

Holloway, S., Arnold, T. & Pretorius, K., 'Tallis Systems Development Procedures and Standards', British Telecom Tallis Systems, 1986.

Hoshi, M., 'Local Network Area Size Optimisation', IEEE Transactions on Communications, Vol COM-33, No.3, 1985, pp199-202.

Hu, T.C., 'A Decomposition Algorithm for Shortest Paths in a Network', Operations Research, Vol 16, 1968, pp91-102.

IBM, 'Personal Computer XT Technical Reference', IBM, 1978.

Johnson, D.B., 'Efficient Algorithms for Shortest Paths in Sparse Networks', Journal of the Association for Computing Machinery, Vol 24, 1977, pp1-13.

Johnson, D.S. & Papadimitriou, C.H., 'Computational Complexity', The Traveling Salesman Problem, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan & D.B. Shmoys (eds.), Wiley, 1985.

Johnsonbaugh, R., 'Discrete Mathematics', Macmillan, 1984.

Kamel, M.S. & Ismail, M.A., 'Computationally Efficient Techniques for Multidimensional Data Clustering', IEEE International Conference on Systems, Man and Cybernetics, Bombay and New Delhi, India, Dec 29 1983-Jan 7 1984, pp564-567.

Karg, R.L. & Thompson, G.L., 'A Heuristic Approach to Solving Travelling Salesman Problems', Management Science, Vol 10, 1964, pp225-248.

Karmarkar, N., 'A New Polynomial Time Algorithm for Linear Programming', Combinatorica, Vol 4, 1984, pp373-395.

Karnaugh, M., 'A New Class of Algorithms for Multipoint Network Optimisation', IEEE Transactions on Communications, Vol COM-24, No.5, 1976, pp500-505.

Karp, R.M., 'Reducability among Combinatorial Problems', Complexity of Computer Computations, R.E. Miller & J.W. Thatcher (eds.), Plenum, 1972, pp85-103.

Karp, R.M., 'On the Computational Complexity of Combinatorial Problems', Networks, Vol 5, 1975, pp45-68.

Karp, R.M., 'The Probabilistic Analysis of some Combinatorial Search Algorithms', Algorithms and Complexity: New Directions and Recent Results, J.F. Traub (ed.), Academic Press 1976, pp1-19.

Karp, R.M. & Steele, J.M., 'Probabilistic Analysis of Heuristics', The Traveling Salesman Problem, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan & D.B. Shmoys (eds.), Wiley, 1985, pp181-205.

Kaufmann, A. & Henry-Labordere, A., 'Integer and Mixed Programming: Theory and Applications', Academic Press, 1977.

Kelly, A. & Pohl, I., 'A Book on C', Benjamin/Cummings, 1984.

Kendall, M.G. & Moran, P.A.P., 'Geometrical Probability', Griffin, 1963.

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

Kleinrock, L., 'Power and Deterministic Rules of Thumb for Probabilistic Problems in Computer Communications', Computer Science Department, University of California, 1979.

Kruskal, J.B., 'On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem', Proceedings of the American Mathematical Society', Vol 7, 1956, pp48-50.

Kuhn, H.W., 'The Hungarian Method for the Assignment Problem', Naval Research Logistics Quarterly, Vol 2, 1955, pp83-97.

Lane, J.E., 'Corporate Communication Networks', National Computing Centre, 1984.

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G. & Shmoys D.B., 'The Traveling Salesman Problem', Wiley Interscience Series in Discrete Mathematics, 1985.

Lin, S. & Kernighan, B.W., 'An Effective Heuristic Algorithm for the Traveling Salesman Problem', Operations Research, Vol 21, 1973, pp498-516.

Little, J.D.C., Murty, K., Sweeney, D.W. & Karel, C., 'An Algorithm for the traveling salesman problem', Operations Research, Vol 11, Nov-Dec 1963, pp972-989.

Mars, P. & Narendra, K.S., 'Routing, Flow Control and Learning Algorithms', First IEE Conference on UK Telecommunications Networks: Present and Future, 2-3 June 1987.

Martin, J., 'Design of Real Time Computer Systems', Prentice Hall, 1967.

Matsui, S., 'A Network Optimisation by Mixed Integer Programming', Nippon Univac Kaisha Ltd., Tokyo, Japan, 1978.

Miller, R.E. & Thatcher, J.W. (eds.), 'Complexity of Computer Computations', Plenum, 1972.

Morisita, M., 'Estimation of Population Density by Spacing Method', Department of Biology, Faculty of Science, Kyushu University, Fukuoka, Japan, 1954.

Muir, A. & Hart, A., 'The Conversion of a Telecommunications Network from Analogue to Digital Operation', First IEE Conference on UK Telecommunications Networks: Present and Future, 2-3 June 1987.

Mulvey, J.M., 'Pivot Strategies for Primal-Simplex Network Codes', Journal of the Association for Computing Machinery, Vol 25, No.2, April 1978, pp266-270.

Murray, W.J., 'The Emerging Digital Transmission Network', British Telecommunications Engineering, Vol 1, Oct 1982, pp166-171.

Ore, O., 'Note on Hamilton Circuits', American Mathematical Monthly, Vol 67, 1960, p55.

Orloff, C.S., 'A Fundamental Problem in Vehicle Routing', Networks, Vol 4, 1974, pp35-64.

Ozdamar, M., 'System X in the Evolving UK ISDN', First IEE Conference on UK Telecommunications Networks: Present and Future, 2-3 June 1987.

Papadimitriou, C.H., 'The Euclidean Traveling Salesman Problem is NP-Complete', Theoretical Computer Science, Vol 4, 1977, pp237-244.

Parker, R.G. & Rardin, R.L., 'Guaranteed Performance Heuristics for the Bottleneck Traveling Salesman', Operations Research Letters, Vol 2, 1984, pp269-272.

Posa, L., 'A Theorem Concerning Hamiltonian Lines', Mag. Tud. Alad. Mat. Kut. Int. Kozl., Vol 7, 1962, pp225-226.

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

Prime, 'System Architecture Reference Guide', Prime Computer Inc., 1981.

Reingold, E.M., Nievergelt, J. & Deo, N., 'Combinatorial Algorithms. Theory and Practice', Prentice Hall, 1977.

Robinson, R.W., 'Enumeration of Colored Graphs', Journal of Combinatorial Theory, Vol 4, 1968, pp181-190.

Robinson, R.W., 'Enumeration of Non-Seperable Graphs', Journal of Combinatorial Theory, Vol 9, 1970, pp327-356.

Rothfarb, B., Frank, H., Rosenbaum, D.M., Steiglitz K. & Kleitman, D.J., 'Optimal Design of Offshore Natural Gas Pipeline Systems', Operations Research, Vol 18, 1970, pp992-1020.

Sahni, S. & Gonzalez, T., 'P-Complete Approximation Problems', Journal of the Association for Computing Machinery, Vol 23, 1976, pp555-565.

Schwartz, M., 'Computer Communication Network Design and Analysis', Prentice Hall, 1977.

Scott, A.J., 'Combinatorial Programming, Spatial Analysis and Planning', Methuen, 1971.

Shapiro, D.M., Algorithms for the Solution of the Optimal Cost and Bottleneck Traveling Salesman Problems', Sc.D. Thesis, Washington University, 1966.

Shamos, M.I., 'Computational Geometry', Ph.D. Thesis, Yale University, 1978.

Sharma, R.L. & El-Bardai, M.T., 'Suboptimal Communications Network Synthesis', Proceedings of the International Conference on Communications, 1970, pp19.11-19.16.

Sloane, N.J., 'Handbook of Integer Sequences', Academic Press, 1973.

Spath, H., 'Cluster Analysis Algorithms', Ellis Horwood, 1980.

Sumner, F.H., 'The Provision of High Performance Computing', Proceedings of the IMA Conference of Simulation and Optimisation of Large Systems, University of Reading, September 23rd-25th, 1986.

Swamy, M.N.S & Thulasiraman, K., 'Graphs, Networks and Algorithms', Wiley, 1981.

Thie, P.R., 'An Introduction to Linear Programming and Game Theory', Wiley, 1979.

Tucker, A., 'Applied Combinatorics' Wiley, 1980.

Turing, A.M., 'On Computable Numbers with an Application to the Entscheidungsproblem', Proceedings of the London Mathematical Society, Series 2-42, 1936, pp230-265.

Upton, I.S.A., 'X-Stream Services', Confidential Communication, 1985.

Upton, I.S.A., 'Using the Plymouth Model with the Indian Network', Confidential Communication, 1987.

Van Der Cruyssen, P. & Rijckaert, M.J., 'Heuristics for the Asymmetric Travelling Salesman Problem', Journal of the Operational Research Society, Vol 29, 1978, pp697-701.

Van Slyke, R. & Frank, H., 'Network Reliability Analysis', Networks, Vol 1, 1972, pp279-290.

Volgenant, T. & Jonker, R., 'A Branch and Bound Algorithm for the Symmetric Traveling Salesman Problem Based on the 1-Tree Relaxation', European Journal of Operations Research, Vol 9, 1982, pp83-89.

Wallstrom, B., 'Methods for Optimizing Alternate Routing Networks', Ericsson Technics, No.1, 1969.

Wherry, A.B., 'Investment Appraisal in Project Planning', Telecommunications Networks, J.E. Flood (ed.), Peter Peregrinus, 1975, pp282-299.

Wilkov, R.S., 'Analysis and Design of Reliable Computer Networks', IEEE Transactions on Communications, Vol COM-20, No.3, June 1972, pp660-678.

Wilson, R.J., 'Introduction to Graph Theory', Longman, 1972.

Wilson, R.J. & Beineke, L.W. (eds.), 'Applications of Graph Theory', Academic Press, 1979.

Yu, W., Majithia, J.C. & Wong, J.W., 'Distributed Routing Algorithm for a Circuit Packet Switching Network', IEE Proceedings, Vol 131, Pt.F, No.7, Dec 1984, pp751-760.

Zahn, C.T., 'Graph Theoretical Methods for Detecting and Describing Gestalt Clusters', IEEE Transactions on Computers, Vol C-20, No.1, January 1971, pp68-86.

PAPERS PUBLISHED/SUBMITTED

1. Grout, V.M., Sanders, P.W. & Stockel, C.T. (a), 'Practical Approach to the Optimisation of Large Scale Switching Networks', Computer Communications, Vol 10, no.1, February 1987, pp30-38.

2. Grout, V.M., Sanders, P.W. & Stockel, C.T. (b), 'Reduction Techniques Providing Initial Groupings for Euclidean Travelling Salesman Patching Algorithms', Submitted to Applied Mathematical Modelling, November 1987.

3. Grout, V.M., Sanders, P.W. & Stockel, C.T., 'Large Scale Telecommunication Network Optimisation', Simulation and Optimization of Large Systems, A.J. Osiadacz (ed.), Oxford University Press, 1988, pp263-283.

4. Grout, V.M. & Sanders, P.W., 'Communication Network Optimisation: Past, Present and Future', Submitted to Computer Communications, February 1988.

OTHER PUBLICATIONS

1. Grout, V.M. (a), 'Plymouth Polytechnic Network Optimisation Package - User Guide', British Telecom Tallis Consultancy, 1987. (included as appendix C)

2. Grout, V.M. (b), 'Plymouth Polytechnic Network Optimisation Package - Software Documentation', British Telecom Tallis Consultancy, 1987. (not included in thesis)

Practical approach to the optimization of large-scale switching networks

V M Grout*, P W Sanders* and C T Stockel† detail the optimization of large-scale switching networks

There are a number of techniques that enable networks to be optimized. Neither search techniques nor heuristic methods are entirely satisfactory in solving practical problems. The authors describe an approach to network optimization that utilizes an algorithm that reduces the number of nodes and consequently the time taken to find a solution. The approach is based on two basic stages: a preparation stage and an optimization stage.

Keywords: communication networks, network optimization, large scale switching networks

In recent years there has been a dramatic increase in the amount of data being transmitted and with the appearance of many new services, such as electronic mail, viewdata and facsimile, it seems that the days of pencil and paper as the primary means of communication are numbered. This demand for communication has led to a similar increase in the number and size of both public and private networks. From the grand scale of the public telephone network, through the variety of private communication systems used by banks, shops etc., to the small system employed by a company in a single building, there is the common need for efficient terminal interconnection.

As the amount of data flowing in various guises increases, the size of new and existing networks increases. Not only is it necessary to increase the traffic carrying capacities of the main routes in such networks but the number and distribution of terminal and switch locations grows and changes. Modem electronic communication may be very flexible and efficient but consequently can be rather expensive. These costs are becoming an increasingly important part of company financial matters with growing amounts of money being put aside for such purposes, so the requirement that any network be connected in the most economical way is of extreme importance. The problem for the network planner is to determine the 'best' way of connecting the terminals together where 'best' is usually taken to mean 'cheapest' with consideration given to constraints on performance under possible failures and heavy traffic conditions.

There already exist a number of methods that may be used to achieve network optimization. Some find the exact optimum by search techniques,[1-3] while others use heuristic methods in order to arrive at an approximation of the optimum in a shorter time[4-8]. None of these is entirely satisfactory in solving the real problem in practice. Many of them take excessive amounts of processing time and/or storage resources, while others make such drastic assumptions and simplifications that the end result bears very little relevance to the practical requirements of the problem. There is clearly a need for a new approach to the problem. A method is required which runs within acceptable limits of time on a smaller machine and operates on realistic networks to give results that are within a specified amount of the true optimum.

The work presented here was initially carried out with regard to telephone networks but this distinction is of no real importance. The principles discussed will apply

*Department of Communication Engineering, Plymouth Polytechnic, Plymouth, Devon PL4 8AA, UK
†Department of Computing, Plymouth Polytechnic, Plymouth, Devon PL4 8AA, UK

equally to any problem that requires that a number of data source/sinks be linked together to form a hierarchical system in such a way as to minimize the total cost. The development of the techniques presented in this paper was achieved using an IBM PC-XT personal computer. The same machine was also used for extensive numerical testing of these techniques.

The results of this work will eventually form part of a suite of programs to be used by British Telecom CFM/NC (Communications and Facilities Management/Network Consultancy).

DESCRIPTION OF THE PROBLEM

Although there is a great variety of network types in computer and telephone systems, they can all be expressed in a common uniform description as follows.

A network N consists of n nodes, where each node i ($i = 1, 2, \ldots, n$) is uniquely defined by its coordinates in the x-y plane. For the general nationwide case, these could be Ordinance Survey grid references. In addition to the nodes there must be at least one $n \times n$ matrix which describes the amount of data (traffic) flowing between each pair of nodes in N. In other words:

$\mathbf{A} = (a_{ij})$ where a_{ij} is the amount of traffic that leaves i and is destined for j.

There may be more than one traffic matrix if there is more than one type of traffic produced in the system (voice, data etc.), but for ease of explanation it shall be assumed that there is only one. The following discussion extends without difficulty to the case where there are two or more matrices. The combination of node locations and traffic matrix completely describes the network N.

An examination of this information will yield further details of the particular network in question. There may be large or small amounts of traffic involved, the nodes may be distributed uniformly or clustered into groups, but for all networks the problem remains the same; namely 'connect up the n nodes in such a way as to minimize the total cost subject to performance constraints'. These constraints will be of reliability, security and grade of service (the percentage of data which fails to transmit). A typical network problem and solution is shown in Figure 1.

Two types of links are currently used in practice. Single analogue circuits suitable for carrying relatively small amounts of speech/data and wideband digital links (equivalent to a number of analogue links) applicable for the higher levels of the network. The nodes shown as squares are transits and are in the best places to switch traffic on the digital links between them. A typical structure would be n network nodes with M of them chosen as transits. Each network node is connected directly to only one of the transit nodes (if a node is a transit then we think of it as being connected to itself) to form M stars. The higher level (intertransit) links are determined separately to achieve the cheapest solution. For two ordinary network nodes to communicate, the traffic must be routed via the path indicated in Figure 1. These intertransit links are usually digital, whereas the

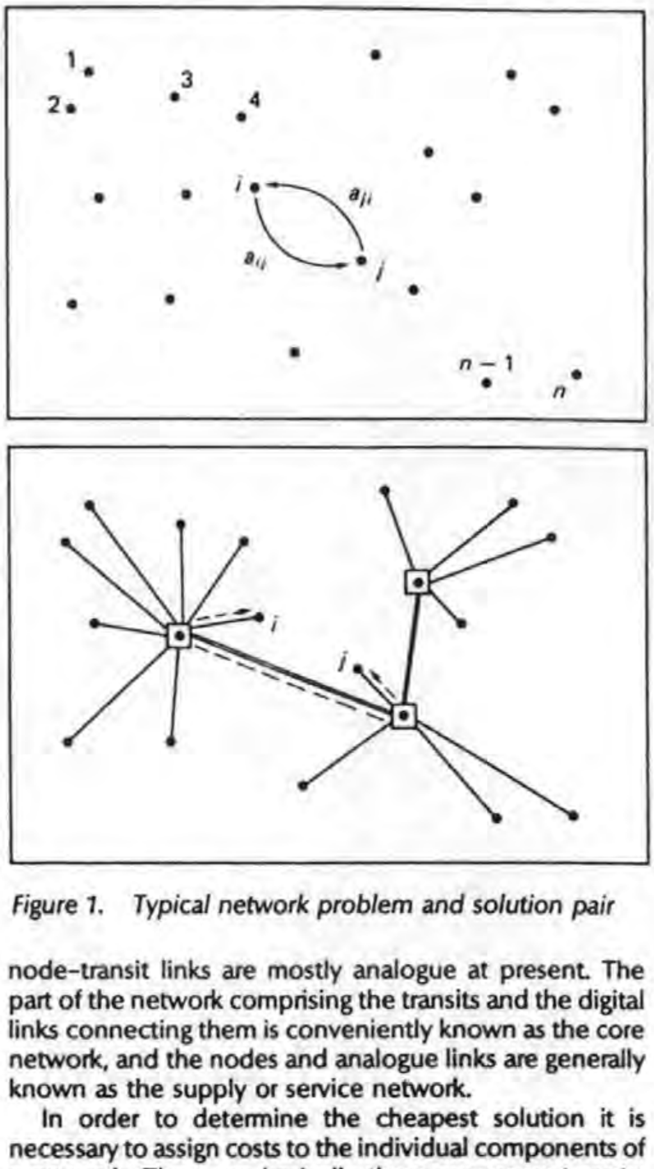


Figure 1. Typical network problem and solution pair

node–transit links are mostly analogue at present. The part of the network comprising the transits and the digital links connecting them is conveniently known as the core network, and the nodes and analogue links are generally known as the supply or service network.

In order to determine the cheapest solution it is necessary to assign costs to the individual components of a network. There are basically three component costs: cost of switches (transits), cost of analogue links and cost of digital links. Therefore, in general, the cost of a network N is given by:

$$C_N = \sum_{\substack{\text{all transits}\\ \text{in } N}} C_T + \sum_{\substack{\text{all analogue}\\ \text{links in } N}} C_A + \sum_{\substack{\text{all digital}\\ \text{links in } N}} C_D$$

where C_T, C_A, C_D are the costs of individual transits, analogue links and digital links respectively.

The cost of a switch can be considered as comprising a fixed installation charge and an additional incremental cost per port, where a port is equivalent to one input/output line entering or leaving the switch. The cost of a single (digital or analogue) line comprises a fixed connection charge plus a cost per unit distance.

$$C_T = a_T + b_{TA}\,p_A + b_{TD}\,p_D$$

where a_T is a standing charge; p_A and p_D are the number of analogue and digital input/output lines entering or leaving the transit switch; b_{TA} and b_{TD} are the costs per analogue and digital input/output port at the transit switch.

$$C_A = a_A + b_A d$$

where a_A is a standing charge; b_A the cost per unit distance and d is the length.

$$C_D = a_D + b_D d$$

where a_D is a standing charge, b_D the cost per unit distance and d is the length.

In general, the optimum solution for a network will have a similar appearance to Figure 1. The use of a partial mesh for the top levels and a set of stars for the low level is particularly suited for communication networks and we make use of this in our optimization. The two level (node-transit) situation is also commonly accepted as opposed to (say) a three level (node-transit-'super transit') configuration.

EXISTING METHODS

In order to determine the optimum arrangement of a network of the form described above, the most obvious method is simply to generate each possible network in turn, determine the total cost of that particular network and compare costs in order to find the cheapest. This, however, is not practical for large arrangements. If there are n nodes in the network and the optimum has M^* transits, we have to let M, the number of transits being 'tried out', range from 1 to n in order to determine the value of M^*. Within each of these loops it is necessary to consider every one of the

$$^nC_M = \frac{n!}{M!(n-M)!}$$

combinations of M transits from n nodes. Also every possible set of intertransit connections must be considered for each combination. This number is equivalent to the total number of graphs that can be constructed on M points, with the exception that half of them will be disconnected and thus give rise to an unacceptable solution. Taking this into account, the number of different sets of intertransit links (or alternatively, the number of possible core networks) for each combination of M transits is:

$$\frac{2^{(M(M-1)/2)}}{2} = 2^{(M(M-1)/2)-1}$$

Consequently, the total number of networks that must be examined in this form of exhaustive search is:

$$\sum_{M=1}^{n} \left[\frac{n!\, 2^{(M(M-1)/2)-1}}{M!\,(n-M)!} \right]$$

Since the summand is exponential in M, the value of the above expression becomes enormous as M increases. However, this expression takes into account only the physical topology of the network; there is no consideration given to how traffic will be routed over the links. If there is no direct link between two transits, which is the best path to take through the network?

The appreciation of the need for simplification has produced a wide variety of network optimization methods. Some of these use established mathematical techniques[2, 5, 8], while others have developed new algorithms for the purpose[1, 4, 7, 9]. One of the commonest approaches is to formulate the network problem as a linear programming task and then use methods such as the simplex method to obtain a solution. In order to do this it is necessary initially to define certain values within the network. The usual formulation is as follows:

Let $y(i)$ ($i = 1, 2, \ldots, n$) be a variable taking the value 1 if a transit is set at node i and 0 otherwise.

Let $x(i, j)$ ($i, j = 1, 2, \ldots, n$) be a variable taking the value 1 if there is a link between i and j, and 0 otherwise.

Let $CT(i)$ be the cost of a transit at node i.

Let $CL(i, j)$ be the cost of a link between i and j.

The problem can then be defined as:

$$\text{Minimize } Z = \sum_{i=1}^{n} CT(i)y(i) + \sum_{i,j=1}^{n} CL(i,j)x(i,j)$$

$$\text{subject to } x(i,j),\ y(i) >= 0 \qquad i, j = 1, 2, \ldots, n$$

$$\text{and} \sum_{i=1}^{n} y(i) >= 1 \qquad \sum_{i=1}^{n} x(i,j) >= 1$$

$$j = 1, 2, \ldots, n$$

which can be solved using an established linear (integer) programming method.

This seems quite satisfactory on the surface but a closer inspection shows that assumptions have been made with the original problem in order to obtain it in this convenient form. The 'constants' $CT(i)$ and $CL(i, j)$ cannot be defined in this way. Referring back to the cost formulas given in the previous section, it can be seen that they are not constants at all. The cost of a transit at site i depends on its size, which in turn depends on where the other transits are in the network and how many there are of them. Similarly it is impossible to state the cost of the link joining i to j since this will depend on how much traffic it has to carry which in turn depends on the final topology of the network. This sort of information is unavailable at the beginning of the optimization process and so cannot be provided as input to a linear programming package. Unfortunately, then, this need for reality reduces the applicability of a large section of the work carried out to date on network optimization.

It is not only the mathematical programming approaches that fall into this trap. Many of the more constructional methods which deal in nodes and links rather than constants and variables also make impractical assumptions. With the increased complication that variable costs cause, the amount of work that needs to be done increases dramatically. The numbers involved are not as large as for the case of the exhaustive search but they are large, certainly increasing as an exponential function of n. Most existing methods run on medium sized or large machines and very few can deal with more than a few hundred nodes even then.

NEW APPROACH

The main problem with the optimization is the number of nodes in the network. The time taken (as well as resources used in general) to produce a solution is a function of that number, $T = f(n)$. The difficulty is that this time becomes too large as n increases. The basic objective, then, in seeking a solution is to decrease the value of T which can be done in two ways:

- Transform the function f to a (less severe) function g. Then the new time T' will be given by $T' = g(n)$ and $T' << T$.
- Transform the number n to a (smaller) number q. Then the new time T' will be given by $T' = f(q)$ and again $T' << T$.

Replacing the function f by g means simplifying the optimization process which in turn means making assumptions and approximations in finding a solution. In contrast, the second possibility requires that work is done on a reduced number of nodes with the same optimization process as before. While literature on network optimization contains many algorithms of the first type, very little has been done with respect to the second. It is our belief that it may be in this direction that a more acceptable solution lies. If somehow the number of nodes under consideration can be reduced before optimization begins in earnest then the corresponding value of T will have been reduced as well. A method is now presented which consists of two basic stages: a preparation stage and an optimization stage. (See Figure 2.)

Having obtained a reduced number of nodes (by whichever method is appropriate) optimization proceeds on the new smaller network approached in three stages, making four in all including the reduction. First, a quick approximation method is used in order to find a reasonably good starting set of transits, then a sequence of perturbations and adjustments on these transit positions is made and finally, when no further improvements are to be found and the final set of transits is established, the best set of intertransit connections (the optimum core network) is found.

The merit of this modular structure is that only as much work as is necessary is carried out at each stage before

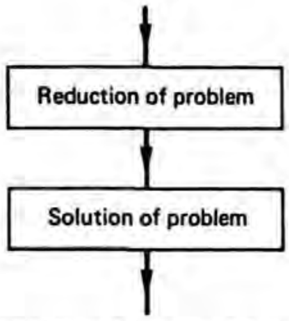


Figure 2. Presentation of a method as two basic stages

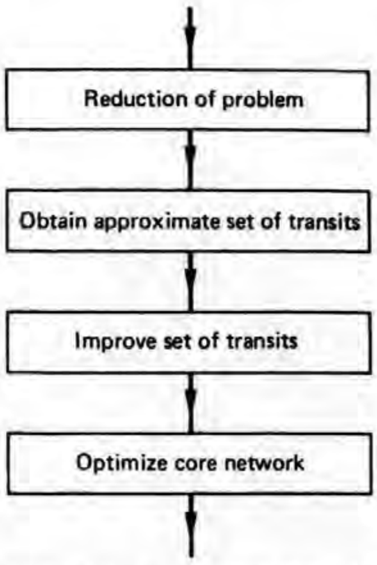


Figure 3. Modular structure of complete method

moving on to the next, but each stage is a little more rigorous and consequently more accurate than the previous one. This method is advantageous compared with existing procedures as it does not try to do everything all at once. The n original nodes are replaced by q representative ones, an initial set of transits is obtained very quickly, this set is modified as far as possible, and finally, with the transits established, the links between them are optimized.

Stage one — reduction

Consider a network with n nodes and an $n \times n$ traffic matrix to describe the traffic flow about the network. The reduction stage is used to determine the position of $q(q < n)$ nodes which represent the original network. In practical terms, q might be 25 or 50 whereas n could be over 1000. This reduction takes place one stage at a time as follows.

By considering all the n nodes in the network, select the two that are closest together. These nodes are then replaced by a single node which represents them, in terms of both their position and traffic characteristics (see Figure 4). The position of this equivalent node is found on the line joining the two points it has replaced, but weighted toward the node with the greater amount of traffic entering and leaving it. Symbolically, if a node r replaces nodes i and j then the x and y coordinates of r are given by:

$$x(r) = \frac{W(i)x(i) + W(j)x(j)}{W(i) + W(j)} \quad y(r) = \frac{W(i)y(i) + W(j)y(j)}{W(i) + W(j)}$$

where $W(i)$ and $W(j)$ are weighting factors indicating the amount of traffic there is at i and j respectively, determined at the beginning of the process. The weighting factor of the new point r, $W(r)$, is calculated as:

$$W(r) = W(i) + W(j)$$

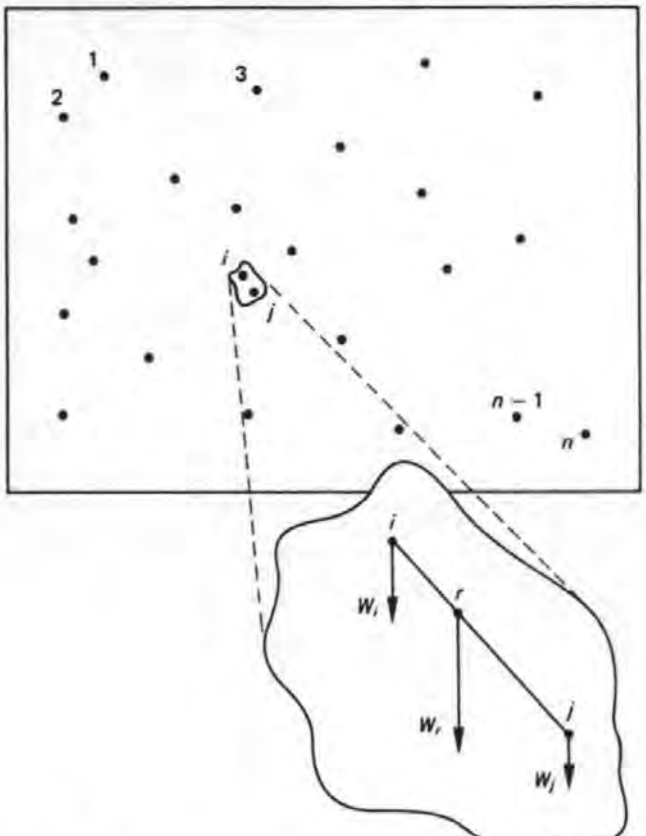


Figure 4. Single stage of reduction

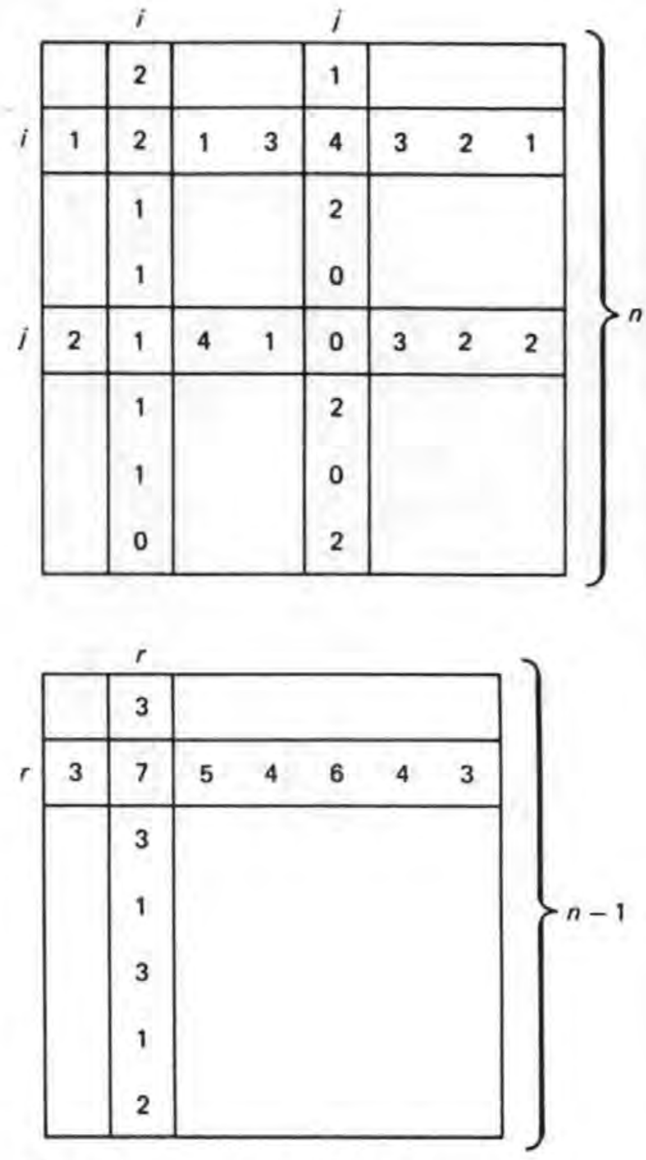


Figure 5. Reducing the traffix matrix

The traffic matrix is updated and reduced by combining the original rows and columns for i and j to give a single row and column for r (see Figure 5).

After this reduction stage we have a network with $n - 1$ nodes and a traffic matrix of size $n - 1 \times n - 1$. The process is then repeated to give an $n - 2$ node network, an $n - 3$ node network and so on. By repeating this $n - q$ times, the required q node network is obtained. These q nodes are located and traffic loaded such that they form a contracted yet accurate picture of the original network. It is important to realize that these are not actual nodes but a representative set on which the optimization may begin.

Stage two — star optimization

We will now find a first approximation to the optimum number of transits and their locations. First, consider what happens when the number of transits is increased. The switch costs obviously increase steadily, the cost of analogue links in the supply network decreases (sharply at first then gradually since the saving in analogue links caused by employing two transits rather than one transit is greater than the saving in analogue links caused by employing five instead of four, for example) and the digital links within the core network increase since there are more transits to interconnect. The result is the curve shown in Figure 6. The overall cost reaches a minimum at $M = M^*$, where M^* is the optimum number of transits, and then begins to increase again. This means that it is no longer necessary to try every value of M from 1 to n in order to find M^*. Instead the best solution with one transit is found, then the best solution with 2, then 3 and so on. As soon as the cost begins to rise again, it is apparent that M^* was in fact the value of M tried previously and therefore it is unnecessary to continue.

For each particular value of M, it is necessary to examine every combination of M transits from q nodes. These sets must be generated in order and tested one at a time. For each set of M transits there may be a very large number of possible core networks — far too many to be examined at this stage, so the assumption that the M transits are to be connected in star formation is made, with one at the centre and $M - 1$ connected to it as shown in Figure 7. This cuts down the number of core networks to be studied from an unmanageable number to just M each time since it must be determined which

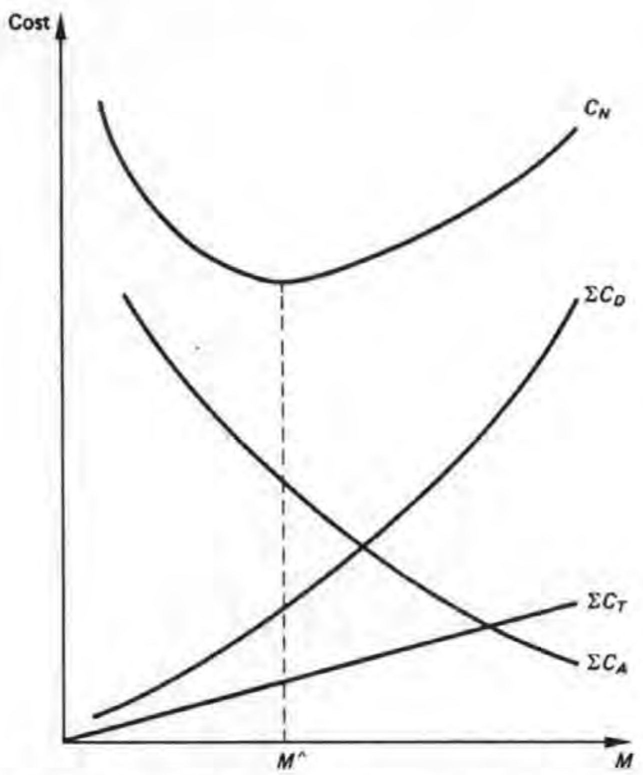


Figure 6. Variation of costs with respect to the number of transits

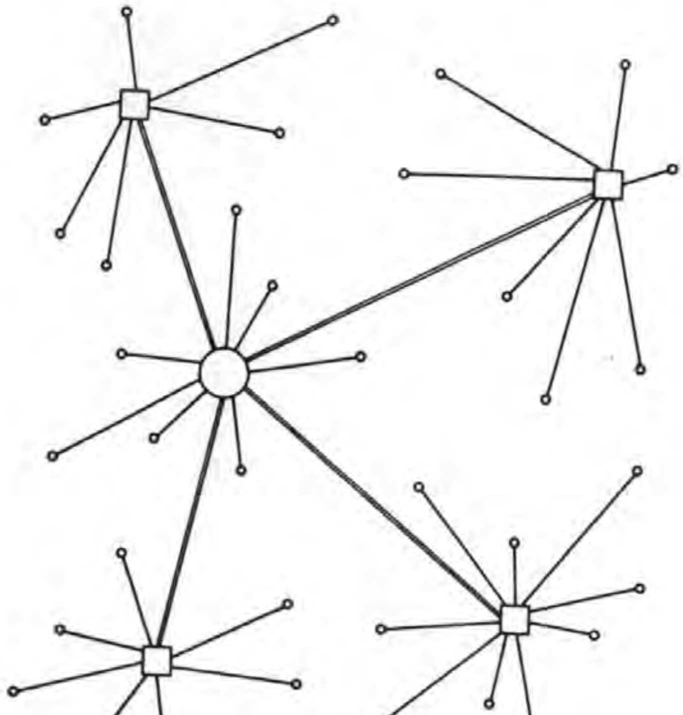


Figure 7. Star configuration core network

transit is best at the centre of the star for that set. This allows the production of a quick starting set of transits without having to search through large numbers of networks. This approximate solution is then passed on to the next section to be improved. The justification for this approach lies in the tendency for real network solutions to have star topologies within the core network or an arrangement that is very similar.

Stage three — transit perturbation

A first attempt at an optimum set of transits now exists which will be a good approximation in some cases but not so in others. The set was obtained with a specific way of connecting the transits together, so by introducing an awareness of different core network possibilities into the proceedings, an attempt to improve upon this for the network in question is made. Each transit in turn is taken from its position and moved gradually around the other nodes in the network. At the same time, different forms of interconnection are tried. If, at any stage, an improvement is encountered, then it is recorded. Having tried out all the different positions for the first node and a number of different connections as well, the choice that gave the greatest improvement is selected and the network is updated accordingly. This is now a (possibly) different set of transits with a (probably) different set of connections from the original star. The entire procedure is now repeated with the second transit, then the third and so on. Having completed the procedure for the *M*th transit, the first transit is selected again and the complete cycle repeated. All of the time, continuously improved sets of transits are obtained by moving further and further away from the initial star.

Eventually, however, a stage will be reached when no further improvement can be found. As soon as one complete cycle of transits is observed without finding any improvement, the procedure is finished. The new set of transits is accepted and passed to the final section to have the links between them optimized.

Stage four — core network optimization

The number and position of the *M* transits are now established. It only remains necessary to determine the most efficient method of connecting the transits together (optimizing the core network). Two methods are presented, one being an extension of the other. The first is very simple and fast but not always as accurate as may be required, while the other is more complicated but can be made as accurate as necessary.

The method begins with all *M* transits connected to one another (a full mesh of $M(M-1)/2$ digital links) and the removal of each link is considered in turn. The removal of a link will result in a saving of cost, but the traffic that was flowing along that link now has to be sent along a different path. This may require additional links in other parts of the network or an increase in the fill of existing digital circuits. Whichever the case, removing the link and redirecting the traffic by the cheapest alternative route is tried and the difference in cost is recorded. This is repeated for all digital links in the core network and the link whose removal caused the greatest improvement is removed permanently from the network, with the traffic being rerouted in the best way possible. This single stage procedure is repeated until either:

- there are no further links whose removal gives an improvement, or
- the removal of any further link disconnects the network.

The core network, and therefore the complete network, is now optimized.

The simple method described is quite sufficient in many cases. However, it can lose accuracy because it makes decisions on a relatively small amount of information at each stage, namely the improvement in cost caused by the removal of a single link. It is possible that a link could be removed early in the process which would turn out to be of great importance toward the end of this process. This, unfortunately, is the bane of all heuristic procedures in one form or another.

Matters can be improved considerably by a 'look-ahead' process in the core network optimization section. In this, it is necessary to consider the removal not just of one link at a time, but of a series of links. Every possible sequence of link removals up to a maximum of *S* stages ahead is examined, and the best sequence, on the basis of the greatest improvement in cost after *S* removals, is selected. However, not all of those links are removed, only the first one in the sequence. The process then repeats from this new core network with stopping criteria exactly as for the simple method. This is very similar to the way in which a chess playing program works. The machine 'looks ahead' as far as possible at each stage, finds the most advantageous board position after a certain number of projected moves, then plays the move that could lead to this position. There is no guarantee that this position will be reached because better information will become available as the game progresses and a different course could be taken.

RESULTS

In order to examine the model in detail, an exhaustive search or full optimization program was developed to run with the same network examples, but on a large Prime 9950 computer. This is a fast, powerful machine in comparison with the IBM PC-XT and is capable of making huge numbers of calculations in a short time. Clearly, such a machine is needed to run routines of this complexity. Even so, exhaustive search methods must be treated with care. Although very slow and requiring cpu times in the order of days and weeks, it produces exact reference networks by trying out every possible network and recording the cheapest. It also gives a measure of the complexity of the job in hand because the program becomes impossible to run for *n* greater than about 30 or 40 depending on the particular characteristics of the network concerned. Comparisons of results between the model and this full optimization program have been very reasonable indeed. Of the 20 or so networks tested simultaneously so far, the overall optimum (number as well as location of transits) was obtained in all but one case (see Network 3 in Table 1). This was an artificial network deliberately constructed in a totally symmetric form with completely uniform traffic flow. Its purpose was to test the model to the extremes of its operation. Even in this instance the difference in cost between the modelled solution and the correct one was less than 1%. If the number of transits was constrained to be artificially higher or lower than the optimum value of M^*, for the purposes of determining the optimum network with a specific number of transits, then the results obtained were exactly correct around 80% of the time with the average error in

Table 1. Six typical network comparisons between the Prime and the IBM PC-XT

Network no	No of nodes	No of transits	Cost of 'optimum' solution produced by: Prime	(units of cost) IBM PC-XT	Error (%)
1	19	1	1.997	1.997	0.0
1	19	2	1.603	1.603	0.0
1	19	3	1.591	1.591	0.0
1	19	4	1.590*	1.590*	0.0
1	19	5	1.668	1.670	0.1
2	50	1	1.532	1.532	0.0
2	50	2	1.219	1.219	0.0
2	50	3	1.199	1.199	0.0
2	50	4	1.190*	1.190*	0.0
2	50	5	1.203	1.203	0.0
2	50	6	1.256	1.256	0.0
3	49	1	2.340	2.340	0.0
3	49	2	2.254	2.254	0.0
3	49	3	2.152	2.152	0.0
3	49	4	2.093	2.093*	0.0
3	49	5	2.079*	2.095	0.8
3	49	6	2.146	2.155	0.4
4	18	1	1.512	1.512	0.0
4	18	2	1.484*	1.484*	0.0
4	18	3	1.509	1.509	0.0
4	18	4	1.562	1.562	0.0
5	25	1	0.965	0.965	0.0
5	25	2	0.952*	0.952*	0.0
5	25	3	1.001	1.001	0.0
5	25	4	1.090	1.122	2.9
6	21	1	1.935	1.935	0.0
6	21	2	1.895	1.895	0.0
6	21	3	1.891*	1.891*	0.0
6	21	4	1.899	1.908	0.4

where* indicates the overall 'optimum' produced by each machine for each of the six networks the remainder of cases being less then 1% and bounded by 5%. These are considered to be excellent results for a program running so much faster than existing methods. It appears that it is only when particularly suboptimal numbers of transits are requested by the user that the model begins to lose accuracy and, even then, to within perfectly acceptable limits. Table 1 gives the results of six typical network comparisons between the Prime and the IBM PC-XT.

The precise time taken by the model to obtain a solution is somewhat flexible. The time taken for the reduction process will depend on the values of n and q but in general it does not constitute a major part of the running time. The speed of the star optimization section will depend on the value of q; the greater the reduction, the smaller the value of q and the quicker the star optimization becomes. The transit perturbation section will run only until there is no more improvement to be found in transit selection and this in turn will depend on the particular network under consideration. The run-time of the core network optimization will be determined by the number of 'look-ahead' stages. Figure 8 shows an example of the saving in time achieved by the reduction process and the assumption of a star core-network at the early stages. It shows graphically the number of different core networks that need to be considered, first by the full optimization method and then by the reduced version. A typical 100 node network with an optimum of

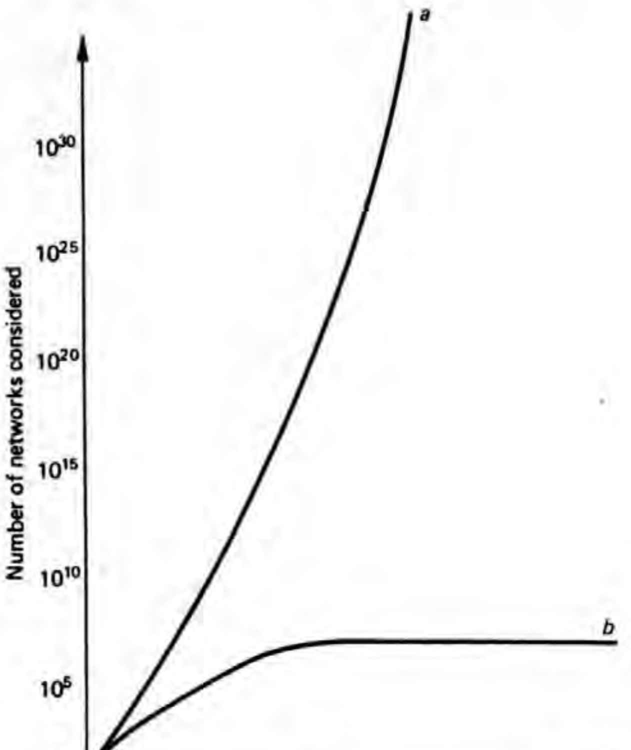


Figure 8. Comparison of computational complexities; $q = 30$ and $M^* = 4$; (a): exhaustive search; (b): model

four transits takes the following approximate times for each section:

```
Reduction — 10 min to q = 30 nodes,
Star optimization — 60 min to M = 4 transits,
Transit perturbation — (on average) 120 min to new M = 4 transits,
Core optimization — one stage 'look-ahead' — a few seconds,
                    two stage 'look-ahead' — 5 min,
                    three stage 'look-ahead' — 60 min,
                    etc.
```

These timings are the approximate average values for each section as measured using an IBM PC-XT personal computer which is a comparatively slow machine. If the program is run on the Prime, which is about 20 times faster, it can be seen that in about 10 min it is possible to obtain an approximation to a solution that was otherwise impossible.

CONCLUSIONS

This is a new and fundamentally different method of optimizing the structure of large switching networks. The results obtained to date are extremely encouraging and have apparently achieved the combined objectives of decreasing the computational work while maintaining the required level of accuracy. It soon becomes impossible to test the model for increasing values of n since full optimization programs take too long to run even on the fastest of machines. Work is proceeding at present to determine the limitations of this technique from a theoretical point of view, and, because no assumptions have been made based solely on communication networks, the application of this technique to many types of network problem seems possible.

ACKNOWLEDGEMENT

The authors gratefully acknowledge British Telecom for their sponsorship of this work and their kind permission to publish this paper.

REFERENCES

1 **Chandy, K M and Russel, R A** 'The design of multipoint linkages in a teleprocessing tree network' *IEEE Trans. on Comput.* Vol C-21 No 10 (October 1972) pp 1062–1066

2 **Matsui, S** *A network optimization by mixed integer programming* Nippon Univac, Tokyo, Japan (1978)

3 **Tabourier, Y** 'All shortest distances in a graph: An improvement to Dantzig's inductive algorithm' *Discrete Math.* Vol 4 (1973) pp 83–87

4 **Karnaugh, M** 'A new class of algorithms for multipoint network optimization' *IEEE Trans. on Commun.* Vol COM-24 No 5 (May 1976) pp 500–505

5 **Hoang, H H** 'Topological optimization of networks: a nonlinear mixed integer model employing generalized benders decomposition' *IEEE Trans. on Auto. Control* Vol AC-24 No 1 (February 1982) pp 164–169

6 **Bahl, L R and Tang, D T** 'Optimization of concentrator locations in teleprocessing networks' *Symp. on Comput. Commun. Networks and Teletraffic* Polytechnic Institute of Brooklyn, New York, USA (4–6 April 1972)

7 **Greenhop, D and Campbell, R** 'Cost 201 — a procedure for the optimization of telecommunication networks' *Br. Telecomm. Eng.* Vol 3 (April 1984) pp 47–58

8 **Mirzaian, A** 'Lagrangian relaxation for the star-star concentrator location problem: approximation algorithm and bounds' *Networks* Vol 15 (1985) pp 1–20

9 **Elias, D and Ferguson, M J** 'Topological design of multipoint teleprocessing networks' *IEEE Trans. on Commun.* Vol COM-22 No 11 (November 1974) pp 1753–1762

REDUCTION TECHNIQUES PROVIDING INITIAL GROUPINGS FOR EUCLIDEAN TRAVELLING SALESMAN PATCHING ALGORITHMS

V.M. Grout, P.W. Sanders and C.T. Stockel

ABSTRACT

Patching algorithms are accepted methods for dealing with large travelling salesman problems. These techniques involve finding an initial set of groupings of the cities and combining them in such a way as to form an approximation to the optimum tour. It is noted that the solution to the corresponding assignment problem does not provide subtours, in the Euclidean case, which are as effective as in the general case. An improvement is then suggested to the usual dissection method involving the step by step reduction of the number of cities.

1. THE GENERAL PROBLEM

The general unconstrained Asymmetric Travelling Salesman Problem (ATSP) is as follows. Given n cities $i=1,2,..,n$ and a cost matrix $C=(c_{ij})$, $i,j=1,2,..,n$, find a cyclic permutation π of the integers $1,2,..,n$ such that

$$z = \sum_{i=1}^{n} c_{i\pi(i)}$$

is a minimum. The cyclic permutation π is known as a tour of the cities $1,2,..,n$.

The ATSP has applications in, among other things, job sequencing problems. The time required to reset a machine to begin job j after completing job i can be represented by c_{ij}. If the machine is to start and finish in the same state then the problem of minimising operation time for n jobs is equivalent to an ATSP of n+1 cities.

The ATSP is known to be NP-complete (Karp [72]) and thus requires heuristic methods to deal with large values of n. One such accepted method involves first solving the associated Asymmetric Assignment Problem (AAP). The AAP requires that any permutation of the integers 1,2,..,n be found so that z, as defined above, is a minimum. In general then a solution to the AAP will consist of a number of subtours and such a solution can be found in $O(n^3)$ steps (Kuhn [55]). These subtours are then patched together according to a fixed system to produce a single tour.

This approach has been found to be quite successful. Much of the accuracy is due to the tendency for solutions to the AAP to consist of large subtours (see appendix). Such a solution requires a small number of patchings to produce a complete tour. The opportunity for error is thus reduced.

2. THE EUCLIDEAN TRAVELLING SALESMAN PROBLEM

The ETSP is a special case of the ATSP. Each city i is defined by its coordinates (x_i, y_i), $i=1,2,..,n$ in the plane. The elements of the cost matrix C are given by $c_{ij} = \sqrt{[(x_i-x_j)^2+(y_i-y_j)^2]}$. A cyclic tour π is then required as before to minimise z.

The ETSP has many applications in practical situations. Vehicle routing, delivery, wiring and machine movement problems can all be transformed to the ETSP, as can almost any problem concerned with minimising toured distance in the plane.

Despite being a restriction of the ATSP, the ETSP is still NP-complete (Papadimitriou [77]). A similar technique can be envisaged for the ETSP as for the ATSP. The solution to the Euclidean Assignment Problem (EAP), however, nearly always consists of subtours of length 2 and 3 (see appendix). This tends to negate the power of any such input to a patching algorithm.

An alternative approach is considered by Karp [76] which exploits the particular characteristics of the ETSP. The appropriate region of the plane is divided into subregions. These subregions provide larger groups of nodes than would be obtained from the corresponding EAP.

Such a technique can, however, encounter problems caused by the way in which these partitions of the plane are determined. This will be a recurrent feature of any such top-down method of division. A bottom-up method would be preferable in which the actual distribution of the cities is used to determine the groupings.

A method of this type can be adapted from Grout, Sanders & Stockel [87]. Starting with the n cities, the following algorithm will replace them by q representative ones.

```
{ Assign a weight $w_i=1$ to each node $i=1,2,..,n$
  $n'=n$
  while $n'>q$ do
      { find $i'$ and $j'$ such that $c_{i',j'}=\min_{ij}[c_{ij}]$
        $n'=n'-1$
        $x_{i'}=(w_{i'}x_{i'}+w_{j'}x_{j'})/(w_{i'}+w_{j'})$
        $y_{i'}=(w_{i'}y_{i'}+w_{j'}y_{j'})/(w_{i'}+w_{j'})$
        $w_{i'}=w_{i'}+w_{j'}$
        for $k = j'$ to $n'$ do
          { $x_k=x_{k+1}$
            $y_k=y_{k+1}$
            $w_k=w_{k+1}$
          }
      }
}
```

This algorithm produces q replacement cities (x_i,y_i), $i=1,2,..,q$ which can then be regarded as the centres of q groups. Each of the original n cities has been included, by the algorithm, in exactly one group. The groupings thus provide an adaptive technique for partitioning the plane prior to any patching algorithm.

3. PATCHING OPERATIONS

There are two distinct types of patching operations which may be used to form complete tours from a number of components. Each has merits dependent upon the particular form of initial solution provided to the algorithm.

3.1. Subtour Combination

If the number of groups is small (with large numbers of cities in each group) and the groups do not necessarily correspond to any Euclidean partitioning, as tends to be the case for AAP solutions, then the most suitable patching technique is one in which two subtours are combined into one at each stage. There are a number of ways in which the two subtours can be chosen such as the largest and smallest or the two closest. The following algorithm chooses the patching that minimises the extra cost incurred at each stage.

Suppose that the number of subtours is q and the permutation given by the AAP is σ.

```
{ $q'=q$
  while $q'>1$ do
       { find $i'$ and $j'$ in different subtours such that
                         $c_{i'\sigma(j')}+c_{j'\sigma(i')}-c_{i'\sigma(i')}-c_{j'\sigma(j')}$
                         $= \min_{ij}[c_{i\sigma(j)}+c_{j\sigma(i)}-c_{i\sigma(i)}-c_{j\sigma(j)}]$
         $q'=q'-1$
         $k=\sigma(i')$
         $\sigma(i')=\sigma(j')$
         $\sigma(j')=k$
       }
}
```

Eventually the q subtours are combined into one.

3.2. Tour Construction

In the Euclidean case the number of groups will be larger and each should correspond to some subregion of the plane. If this is so, the large number of subtour combinations required by the algorithm in section 3.1 may cause such a method to become somewhat unreliable. Moreover it seems reasonable to try to take advantage of any Euclidean interpretation to the groupings if such a structure exists. A more appropriate method is to form a tour which visits each of the groups in turn, find either a tour or a path through all of the cities in each group and patch each one into the larger tour.

A variation of this is given by the following algorithm, adapted from Karp [76]. Assume the appropriate region of the plane has been divided into q subregions R_j, $j=1,2,..,q$ either arbitrarily or via the node reduction algorithm of section 2.

```
{ for j = 1 to q do
    { construct an optimal or near optimal tour of all nodes $i \in R_j$
    }
  for j,k = 1 to q do
    { define $d_{jk}=\min_{i \in Rj\ i' \in Rk}[c_{ii'}]$
    }
  construct the minimum spanning tree (MST) over the subregions $R_j$,
                                                    using the distances $d_{ij}$
  construct a closed walk which includes every subtour and which
                                  passes from one subregion to another by
                                  using each MST edge twice
  transform the closed walk into a tour
}
```

4. RESULTS

The method of reduction for the ETSP presented in section 2 produces initial city groupings of the form provided by both the solution of the EAP and the application of fixed dissection. It is the purpose of this section to examine the accuracy of these groupings. In comparison with the EAP method the subtour combination algorithm of section 3.1 was applied to each set of groupings whereas for the fixed dissection case the tour construction approach of section 3.2 was employed.

4.1. Solutions Via the Assignment Problem

Sets of 25, 30, 40 and 50 cities were generated at random in the plane. 100 such sets were produced in each case. Initial subtours were produced

(a). by solving the EAP.

(b). by the reduction method of section 2.

An upper bound, β was placed upon the number of cities in any of the q groups in option (b) to ensure that the optimum tour could be found for each group. Reduction continued until the point at which β would have been exceeded. Thus β actually determines the value of q. The subtour combination algorithm of section 3.1 was then employed in both cases.

The difference in results between the two methods was slight. Improvements were obtained in about 70% of all cases tested with an average saving of 2-3%. The average improvement (%) of the reduction over the AP method for each value of n and β is shown in table I.

n	β = 3	4	5	6	7	8	9	10
25	0%	0%	0%	1%	1%	1%	1%	2%
30	0%	0%	1%	1%	1%	2%	2%	3%
40	0%	0%	1%	1%	2%	2%	3%	4%
50	0%	1%	1%	2%	2%	3%	4%	5%

Table I

4.2. The Fixed Dissection Algorithm

The same sets of cities were used as in section 4.1. A initial grouping of the cities was obtained via

(a). the fixed dissection method.

(b). the reduction method of section 2.

These groups of cities were then combined into a tour using the algorithm of section 3.2.

On average the fixed dissection method gives improvements of about 5% over the EAP method, depending on the number of cities and partitions of the plane. The average improvement (%) of the reduction method over the fixed dissection method for each value of n and β is shown in table II.

n	β = 3	4	5	6	7	8	9	10
25	0%	0%	0%	0%	0%	1%	1%	1%
30	0%	0%	0%	1%	1%	1%	1%	2%
40	0%	1%	1%	1%	1%	2%	2%	3%
50	1%	1%	1%	2%	2%	3%	3%	3%

Table II

5. CONCLUSIONS

There is clearly some advantage in using this method of reduction to partition the plane prior to an appropriate patching algorithm. As can be seen by the comparisons with the AP method, this patching strategy is of considerable importance and an inappropriate method can affect results to a large extent.

However, when an effective method is used this reduction provides very good groupings of the cities. Not surprisingly, these results improve as the maximum group size increases.

APPENDIX : THE n x n ASSIGNMENT PROBLEM

The input to an n x n Assignment Problem (AP) is an n x n cost matrix $C=(c_{ij})$, $i,j=1,2,..,n$ where, for the purposes of this paper, $c_{ii}=\infty$, $i=1,2,..,n$. The solution is a permutation σ of the integers $1,2,..,n$ which minimises

$$z = \sum_{i=1}^{n} c_{i\sigma(i)}$$

In general, σ is a set of subtours of the cities $1,2,..,n$ although the constraints on the diagonal elements of C serve to eliminate subtours of length one.

A.1. The Asymmetric Assignment Problem

For the AAP no restriction exists on the non-diagonal elements of C. Consider the c_{ij} as drawn randomly from the uniform distribution over some interval [a,b]. Consider any individual city i. The probability that i lies in a subtour of length m is

$$\frac{n-2}{n-1} \times \frac{n-3}{n-2} \times \frac{n-4}{n-3} \times \ \ldots \ \times \frac{n-m+2}{n-m+3} \times \frac{n-m+1}{n-m+2} \times \frac{1}{n-m+1}$$

$$= \frac{1}{n-1} \qquad m=2,3,\ldots,n$$

The expected length of any randomly selected subtour is therefore given by $\underline{m} = (n+2)/2$ which is over half the number of cities. Karp & Steele [85] generated 10 random permutations of 1000 elements (allowing subtours of length one) in which the mean length of the longest cycle was 700.

A.2. The Euclidean Assignment Problem

The EAP has elements of C defined as

$$c_{ij} = \begin{cases} [(x_i-x_j)^2+(y_i-y_j)^2]^{1/2} & i<>j \\ \infty & i=j \end{cases}$$

Shapiro [66] states that solutions to the EAP usually consist of subtours of length two except where odd numbers of of cities force the existence of a single subtour of length three. This is not entirely true. Any solution to the EAP on any number of cities will, usually consist of a mixture of subtours of length two and three (Fig 1 for example).

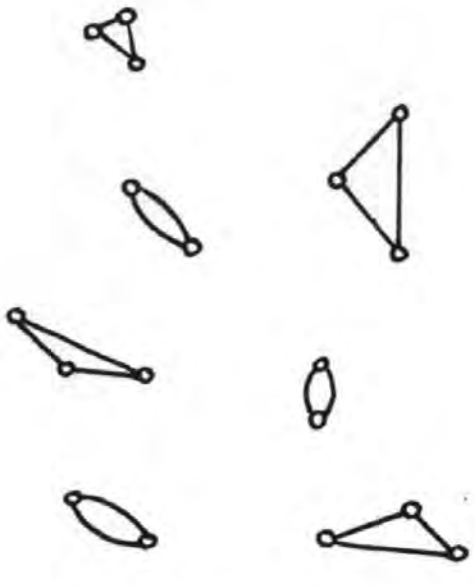


Figure 1

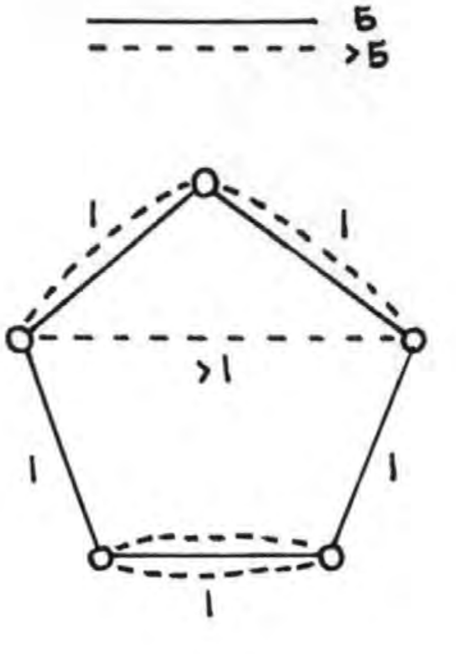


Figure 2

Subtours of length greater than three can exist on groups of odd numbers of cities in extreme cases (see Fig 2) but no complete tour on a group containing an even number of nodes can ever be shorter than the shortest set of subtours of length two as the following shows.

Suppose the number of nodes in any group is even. Let (a,b), (c,d), ..., (y,z) be the set of subtours of length two of least cost so that

$$2c_{ab}+2c_{cd}+\ldots+2c_{yz} =< 2c_{a'b'}+2c_{c'd'}+\ldots+2c_{y'z'}$$

for all other sets of subtours of length 2.

ie $c_{ab}+c_{cd}+\ldots+c_{yz} =< c_{a'b'}+c_{c'd'}+\ldots+c_{y'z'}$ (A.2.1)

Now suppose there exists a tour (a",b"), (b",c"), (c",d"), .., (y",z"), (z",a") of smaller cost such that

$$c_{a''b''}+c_{b''c''}+c_{c''d''}+\ldots+c_{y''z''}+c_{z''a''} < 2c_{ab}+2c_{cd}+\ldots+2c_{yz} \quad \text{(A.2.2)}$$

Equation A.2.1 gives $c_{a''b''}+c_{c''d''}+\ldots+c_{y''z''} >= c_{ab}+c_{cd}+\ldots+c_{yz}$
and $c_{b''c''}+c_{d''e''}+\ldots+c_{z''a''} >= c_{ab}+c_{cd}+\ldots+c_{yz}$

Substituting into equation A.2.2 gives

$$2c_{ab}+2c_{cd}+\ldots+2c_{yz} < 2c_{ab}+2c_{cd}+\ldots+2c_{yz}$$

So no such tour can exist.

REFERENCES

Grout, V.M., Sanders, P.W. & Stockel, C.T., 'Practical Approach to the Optimisation of Large Scale Switching Networks', Computer Communications, Vol 10, No.1, 1987, pp30-38.

Karp, R.M., 'Reducability among Combinatorial Problems', Complexity of Computer Computations, R.E. Miller & J.W. Thatcher (eds.), Plenum, 1972, pp85-103.

Karp, R.M., 'The Probabilistic Analysis of Some Combinatorial Search Algorithms', Algorithms and Complexity: New Directions and Recent Results, J.F. Traub (ed.), Academic Press, 1976, pp1-19.

Karp, R.M. & Steele, J.M., 'Probabilistic Analysis of Heuristics', The Traveling Salesman Problem, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan & D.B. Shmoys (eds.), Wiley, 1985, pp181-205.

Kuhn, H.W., 'The Hungarian Method for the Assignment Problem', Naval Research Logistics Quarterly, Vol 2, 1955, pp83-97.

Papadimitriou, C.H., 'The Euclidean Traveling Salesman Problem is NP-complete', Theoretical Computer Science, Vol 4, 1977, pp237-244.

Shapiro, D.M., 'Algorithms for the Solution of the Optimal Cost and Bottleneck Traveling Salesman Problems', Sc.D. Thesis, Washington University, 1966.

AUTHORS

Vic Grout & Peter Sanders,

Department of Communication Engineering,

Colin Stockel,

Department of Computing,

Faculty of Technology,

Plymouth Polytechnic,

Drake Circus,

PLYMOUTH,

Devon, PL4 8AA, UK.

LARGE SCALE TELECOMMUNICATION NETWORK OPTIMISATION

V.M. Grout and P.W. Sanders
(Department of Communication Engineering, Plymouth Polytechnic)

and

C.T. Stockel
(Department of Computing, Plymouth Polytechnic)

The design of least cost telecommunication networks is an important part of mainstream combinatorial optimisation. Such problems typify the characteristics and general difficulties of the subject as the number of possible solutions is sufficiently large to prohibit exhaustive search techniques. Approximation methods, to date, have achieved only limited success, mainly due to the rather impractical assumptions made in their analysis.

This paper presents a technique for reducing a starting network of unmanageable size down to a smaller representative one. Any solution obtained for the smaller network can then be translated into a suitable solution for the initial network. Such a solution is obtained in stages. Firstly, an initial solution is found via the assumption of a star network connection topology. A series of local perturbations of this starting solution are then effected until no further improvement in cost can be achieved. Finally, a look ahead technique is employed to determine the best connection topology and routeing strategy for the final network.

Results are extremely encouraging and it appears that this method may be far more general than its presentation in this context suggests. This work has been carried out in collaboration with British Telecom CFM.

1. INTRODUCTION

1.1 Background

Telecommunication networks are one of the most important national assets of a modern society. The need for an efficient system at the most economic cost is of paramount importance.

As a consequence, optimisation of the network becomes essential. There is an increasing number of private networks being brought into service to satisfy the diverse communication requirements of large corporations, banks, stores etc. The information being conveyed ranges from speech, through database services, to electronic funds transfer. In general, these systems necessitate that a, possibly large, number of terminal devices are able to communicate with one another, requiring some form of interconnection strategy to satisfy their design objectives. The terminal devices can be regarded as sources or sinks of communication traffic and are known, with reference to graph theory, as nodes. For any node pair (i,j) a value can be determined that represents the amount of a particular type of traffic leaving i and being destined for j. The problem to be solved is then to produce the optimum topological structure for connecting all of the nodes together and to determine the best route or path along which each node pair communicate. Optimum, in this case, can be taken to mean cheapest, subject to constraints such as quality of service and alternate routeing. These requirements specify that no more than a certain fraction of all originating calls are lost due to blocking (insufficient line or switch capacity) or that at least a certain number of nodes or links must fail before the network becomes disconnected.

1.2 Network structure

The particular form of a network solution depends on the individual application for which the network is required. Electronic mail or airline booking systems, where connection to a central computer only is needed, will be satisfied by solutions of the form discussed by Chandy & Russell (1972) and Kershenbaum & Chou (1974). These present an exact and an approximate method respectively for obtaining such a solution. In general, however, communication between pairs of nodes is required and these tree structures become less acceptable since there is only a single route between any node pair, making the network insecure. The more usual form of a solution is shown in Fig. 1 and it is topologies of this type which are considered in this paper.

Here there are two levels to the network. A selected subset of the set of nodes is chosen to act as key nodes or transits, collecting and distributing (switching) the traffic around the network. Each node that is not a transit is directly connected to its nearest transit such that each group thus formed, takes on a star configuration. The transits can be interconnected in any way ranging from a minimal spanning tree to a fully connected mesh subject to the degree of alternate routeing required.

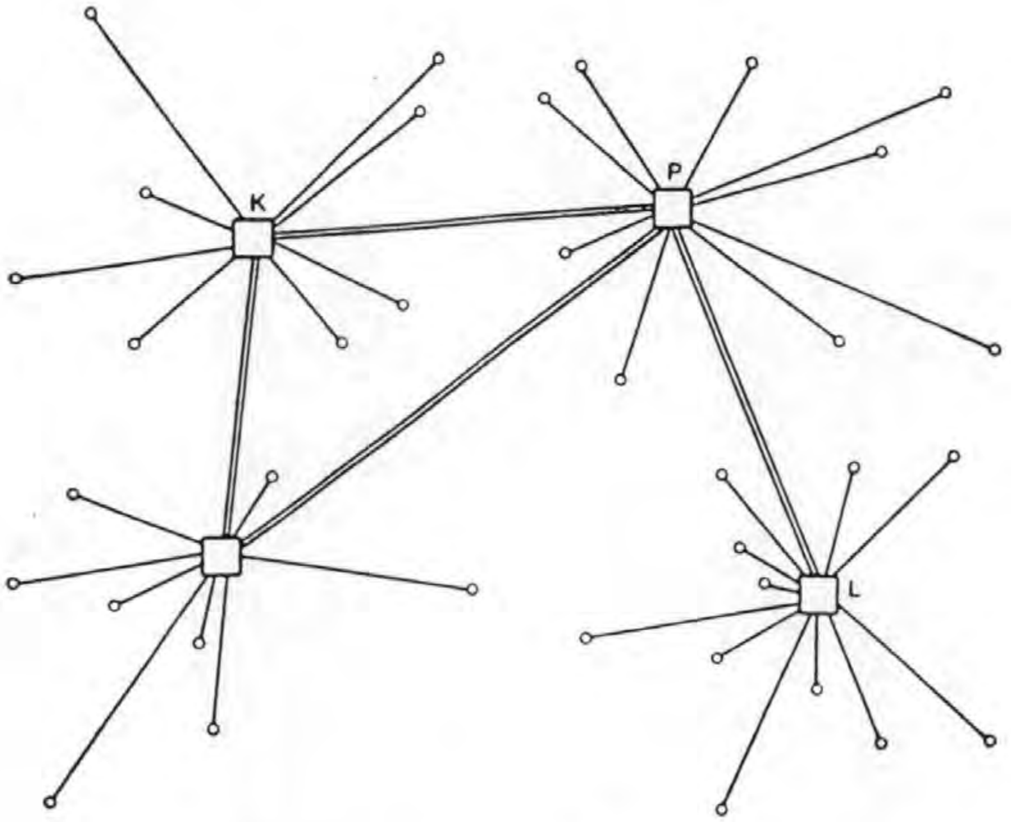


Fig. 1 Common network structure

There are two types of communication link which can be used to connect transits and/or nodes together. The intertransit connections, generally requiring high traffic capacity, always use digital links (standard 30 channel pcm systems) whereas the, often low capacity, node to transit connections can use single analogue circuits or digital systems. In the future, all links will be digital but at present, individual analogue links can be cheaper for small amounts of traffic. The network comprising the transits and the digital links connecting them is often referred to as the transit or core network and the nodes together with the analogue or digital links connecting them to their nearest transit, form what is known as the local or service network. A digital circuit can carry an amount of traffic equivalent to a number of analogue circuits (typically 30) but at a much reduced cost per circuit. The costs of an analogue link, a digital link and a transit switch are respectively given by

$$C_A = a_A + b_A d \qquad C_D = a_D + b_D d \qquad C_T = a_T + b_T p \tag{1.2.1}$$

where a_A, a_D and a_T are installation charges, b_A and b_D are costs per unit distance, d is the length of circuit, b_T is the connection cost per input/output port and p is the required number of such ports. The total cost of any solution network is thus given by

$$C = \sum_{\mathcal{A}} C_A + \sum_{\mathcal{D}} C_D + \sum_{\mathcal{C}} C_T \qquad (1.2.2)$$

where $\mathcal{A}$, $\mathcal{D}$ and $\mathcal{C}$ are the sets of analogue links, digital links and transit switches in the network. The general objective is to minimise the value of C is any particular case.

1.3 The problem

The input to any network optimisation problem (NOP) consists of the locations of n network nodes, (x_i, y_i), i=1,2,..,n, together with a description of the traffic flowing between them. This traffic is represented by one or more matrices of the form $A_\mu = (a_{\mu ij})$ $(i,j=1,2,..,n,\ \mu = 1,2,..,\pi)$ where there are π different types of traffic flow (voice, data etc.) and $a_{\mu ij}$ represents the amount of traffic of type μ originating at i and destined for j.

The problem is then to determine the cheapest connection strategy of the form shown in Fig. 1. The output required is as follows.

- (a) The optimum number of transits ($M^\wedge$)
- (b) The optimum selection of $M^\wedge$ transits from n nodes
- (c) The size of the link from each node to its transit
- (d) Whether this link is digital or analogue
- (e) The core network connection topology
- (f) The size of each intertransit digital link in the core network
- (g) The traffic routeing strategy

This minimisation is to be achieved subject to performance constraints, namely

- (a) Grade of service (GOS)
- (b) Security against failure
- (c) Degree of rerouteing (DOR)

The grade of service (GOS) is defined to be the fraction of all generated calls that are lost in the network due to saturation of facilities. For the purposes of this paper, it suffices to note that, for a particular GOS, the number of analogue circuits (c) required to carry an amount of traffic (A) over a link can be determined from tables based on the Erlang B formula.

$$GOS = \frac{\frac{A^c}{c!}}{\sum_{x=0}^{c} \frac{A^x}{x!}} \qquad (1.3.1)$$

The security of a network is dependent on its connectivity and the number of edge/node disjoint paths between two transits. It is a measure of how many nodes/links must fail before communication between any node pair becomes impossible.

If there is no direct link between two transits in a network then any traffic from one to the other has to be rerouted via another one, two, ... extra transits. The maximum number of such extra transits in any main path is called the degree of rerouting (DOR). For example, in Fig. 1 the DOR is 1, since the maximum length path is between transits K and L which passes through transit P.

2. ANALYSIS OF THE PROBLEM

2.1 Complexity of the problem

The first step in the analysis of the problem is to determine the total number of feasible solution networks which can be constructed on n nodes. Since the solution may involve 1,2,3,...,n transits, the total number of possible solution networks will be of the form

$$N_n = \sum_{M=1}^{n} (\;....\;) \qquad (2.1.1)$$

Now suppose that the number of transits in the solution network is fixed at M. The number of such combinations of M transits from n nodes is

$$\binom{n}{M} = \frac{n!}{M!\,(n-M)!} \qquad (2.1.2)$$

The size of each of the n-M node to transit links is dependent upon the amount of traffic which each node needs to transit to its parent transit. Each link may be analogue or digital, giving 2^{n-M} choices in all.

With the number and combination of transits established, the numbe rof valid core networks which can be constructed on these transits can be found. Assuming that no DOR is defined, this is equal to the total number of connected graphs which can constructed on M nodes, i.e. $2^{(M(M-1)/2)-1}$.

The number of possible routeing strategies is dependent upon the particular topologies of the core networks. For the purposes of enumeration, it will be assumed that such a strategy can be determined along with the core network topology. If this is the case, the total number of feasible networks which can be constructed on n nodes is given by

$$N_n = \sum_{M=1}^{n} \frac{n!\,2^{(M(M-1)/2)-1}\,2^{n-M}}{M!\,(n-M)!} \qquad (2.1.3)$$

2.2 The search for a solution

A standard combinatorial optimisation problem which is known to be NOP complete is the Travelling Salesman Problem or TSP (see Lawler et al. (1985)). The number of possible tours which can be constructed on n cities in the TSP is (n-1)!. Calculating the number of possible solutions for both the TSP and the NOP for the first few values of n, shows that, for a problem of size n, the number of solutions for the NOP increase even faster than for the TSP. Whereas an exhaustive search algorithm for the TSP would take O(n!) steps, such a method for the NOP would take more than $O(2^{n^2})$. Other exact solution methods such as branch and bound (Chandy and Russell (1972)) or linear/integer programming (Matsui (1978)) are available but, though making significant, and sometimes, unrealistic simplifications to the problem still take an unacceptable amount of time as n increases. Clearly, as is the case with the TSP, approximation algorithms are required to deal with the larger problems.

There have been a considerable number of proposals over the years for solving selected parts of the NOP (Gunnarsson & Nivert (1984), Hansler et al. (1972) and Hoshi (1985)) and there are equally many techniques available for dealing with certain simplifications of it (Karnaugh (1976), Tang et al. (1978) and Mirzaian (1985)). Most operate by restricting the possible solution space to that subspace where it is believed, often for intuitive reasons, the solution is likely to lie. This approach can be somewhat unreliable in practice.

To solve the general unconstrained problem, a common approach, in practical situations, is to use a two step heuristic process. This can be regarded as an extension of the drop algorithm first proposed by Bahl and Tang (1972).

This algorithm is shown diagrammatically in Fig. 2. It employs a form of double drop technique whereby transits are systematically removed from the network in such a way as to maximise the improvement in cost. At each stage, the optimal connection strategy for that particular combination of transits is approximated by a sequence of link removals, again working on the principle of least cost within each step.

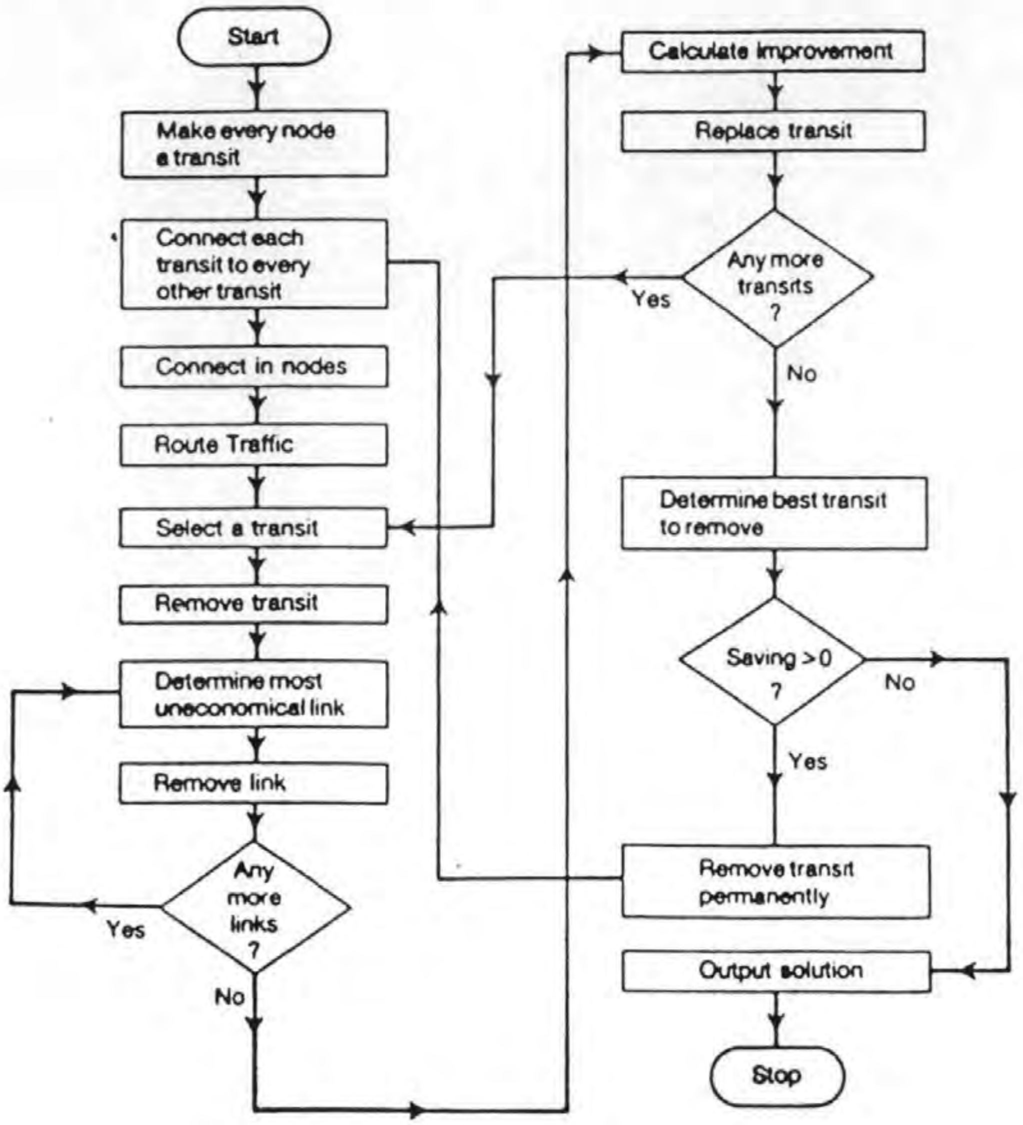


Fig. 2 Double drop algorithm

If implemented efficiently, this technique can deal with several hundred nodes but is entirely a step by step process. It is possible, and in practice quite common, for a node to be eliminated at an early stage which should actually form a transit of the optimum network. This problem can be partly offset by using the method a number of times with different starting conditions.

3. A NEW SOLUTION TO THE PROBLEM

3.1 Overview

The problem outlined in section 1.3 can be solved heuristically in sections. Presented with a set of nodes, the first task is to find the optimum number and location of a subset of these to act as transits. With n nodes and M transits, the number of possible subsets is

$$\frac{n!}{M!(n-M)!} \qquad (3.1.1)$$

Even for a relatively small network with 100 nodes and 10 transits, this is over 10^{13}, which is far too large to allow each to be tested by even the largest machines.

Consider the problem more generally. An algorithm is required to produce a solution to a problem with n nodes. The algorithm will take a time T to operate, where T=f(n). The function f depends on the particular algorithm in use. The problem with any exact search method is that T becomes far too large as n increases so that some form of simplification must be made. This can be achieved in exactly two ways.

(a) Transform the function f to a less severe function g where $T = f(n) >> g(n) = T'$.

(b) Replace the parameter n by a smaller number q so that $T = f(n) >> f(q) = T'$.

where T' is the time taken to solve the new problem in each case. All existing heuristic methods are of the first type. The optimisation process is simplified by a series of assumptions and approximations. This paper presents a new method of the second type in which exact search techniques can still be used (although in practice are not) but on a reduced number of nodes that represent the original network.

This is achieved using a step by step reduction process presented in section 3.2. The value of q, to which the number of nodes is reduced will depend on the efficiency of the optimisation process which is then used. Whilst it would be possible to make q small enough for an exhaustive search technique as suggested in section 2.2 to be used, greater efficiency, in terms of the time/accuracy trade off, can be achieved with a compromise. In this method, reduction is used to replace the large n node network by a smaller q node network which is still too large to be optimised by exhaustive search.

Once a reduced network has been produced, an initial set of transits is derived, relatively quickly, by assuming certain characteristics about the core network. A series of perturbations (local transformations) is then carried out with these transits to improve the network arrangement.

The optimum core network connection strategy and the best way of routeing the traffic around the network is then achieved in one process by initially fully interconnecting the core network and using a flexible look ahead method of link removals.

The necessary size of each analogue and digital link can be found from traffic tables or implicitly from equation (1.3.1) The type chosen for the individual service network links is dependent on the relative costs determined by the required traffic capacities and length. The overall flow of the algorithm is then in five major distinct stages, and is presented here in this form.

3.2 Reduction

Consider the initial network of n nodes described by (x_i,y_i) : $i=1,2,..,n$. With each node i can be associated a weight w_i which is related to the amount of traffic destined for, and being generated by, the node i. The two nodes which are closest together of all node pairs in the initial network are selected, i.e. nodes i and j such that

$$D_{ij} = \sqrt{\{(x_i-x_j)^2 + (y_i-y_j)^2\}} \qquad (3.2.1)$$

is a minimum; i and j are now to be replaced by a single node r, with characteristics which represent i and j, both in terms of position and weights. The new values x_r, y_r and w_r are given by

$$x_r = \frac{w_ix_i+w_jx_j}{w_i+w_j},\ y_r = \frac{w_iy_i+w_jy_j}{w_i + w_j},\ w_r = w_i+w_j \qquad (3.2.2)$$

This reduction stage is shown in Fig. 3. Each traffic matrix must be reduced by one row and one column as well. This is achieved by adding together the entries in each column of row i to the corresponding entry in row j and placing the result in row i, now labelled r. The same is done for the columns. Fig. 4 gives an example. The new node r now represents the original nodes i and j in every sense.

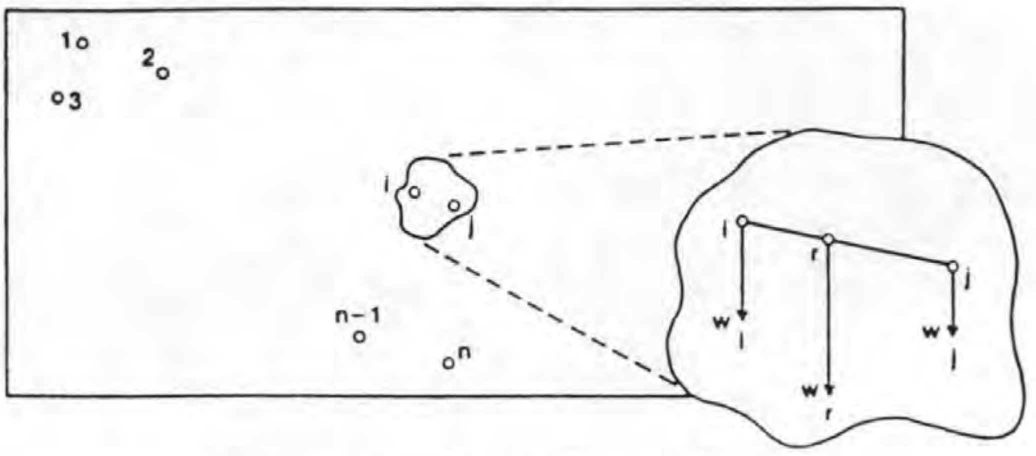


Fig. 3 Single reduction step

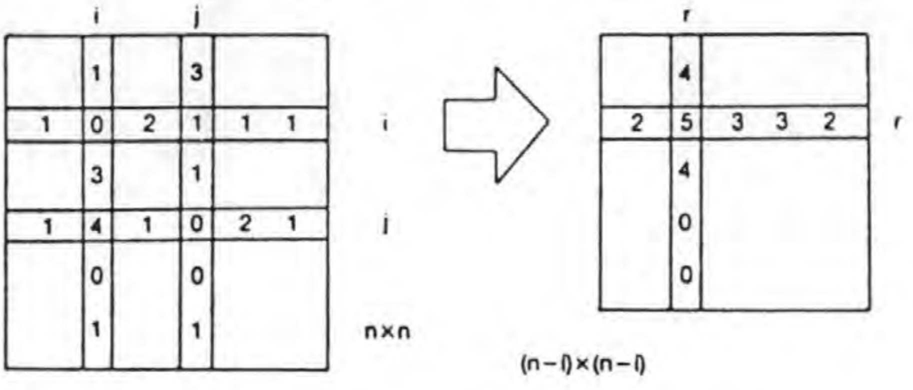


Fig. 4 Reduction of the traffic matrix

A new network has now been produced with n-1 nodes, one fewer than before. If this process is repeated n-q times, a q node network is produced which represents the original in terms of traffic distribution characteristics. These q nodes have been drawn, by the weights, to areas of heavy traffic. The process of transit selection can now take place from this reduced number of nodes. Since these q nodes represent the original n, any combination of M transits selected from these q will also prove to be a good selection to apply to the original network.

Clearly, this weighted reduction process produces replacement nodes at positions which do not correspond to the locations of any of the original nodes. Although these could be used in the next stages of the procedure, the nearest real site is chosen for each transit position.

3.3 Star optimisation

An awareness of the way in which the cost of various networks change will help simplify the task of choosing the transits. As seen in equation (1.2.2), the total cost of any network is comprised of the three components shown in Fig. 5. These are

(a) Service Link Costs. Such costs decrease as the number of transits increases since more traffic is being switched on the digital links between transits.

(b) Core Link Costs. Such costs increase as the number of transits increases for the same reason as in (a).

(c) Transit Costs. Obviously, these costs increase with the number of transits.

The additive results of two steadily increasing functions and and one decreasing function tends to the convex (downwards) curve shown in Fig. 5 (although, in practice, the curve is stepped). This point can be of considerable use when attempting to find the optimum number of transits. If M is a variable denoting a particular number of transits and M^* is the optimum value of M then it is unnecessary to test M=1, M=2, .. , M=q, to find M^*. It is only necessary to continue until the cost begins to rise again. It can then be assumed that the optimum has been passed.

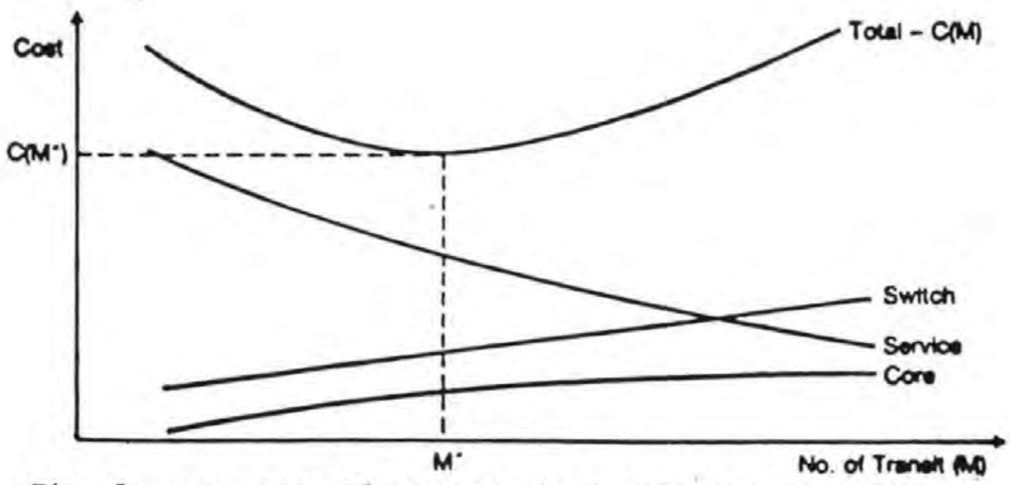


Fig. 5 Separation of cost function into its components

For any particular value of M there are

$$\frac{q!}{M!(q-M)!} \qquad (3.3.1)$$

combinations of transits to be considered. For each of these combinations, there are $2^{(M(M-1)/2)-1}$ possible core networks. For increasing values of M, it becomes impossible to try every one. A study of a number of practical telecommunication networks reveals a strong tendency for the core network to take on the appearance of a star or very close to it in an unconstrained situation. Moreover, at this stage, only an initial solution is required to pass to the perturbation section. With this in mind, it is assumed, protem, that the transits are to be connected together as a star as shown in Fig. 6. There are M such stars for each selection of transits, so there are

$$\frac{Mq!}{M!(q-M)!} = \frac{q!}{(M-1)!(q-M)!} \qquad (3.3.2)$$

different networks to be tested. This part of the problem is now reduced to manageable proportions. Whilst the simplification of the core network to a star is not necessarily justified in every case, this technique will provide a starting solution which has been found good enough, in practice, to present to the next stage.

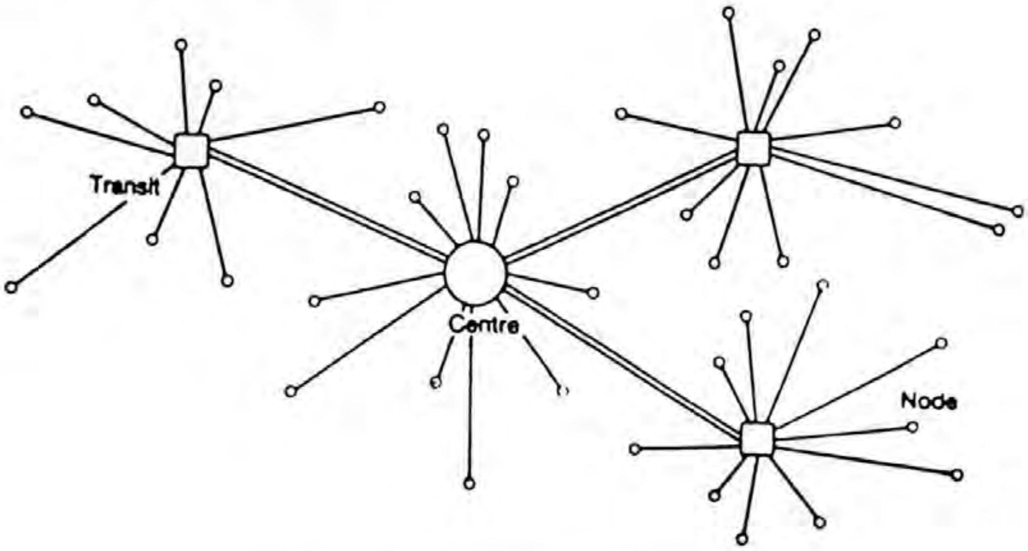


Fig. 6 Star core network

3.4 Transit perturbation

This stage applies small perturbations to the network in order to test for improvement. This is achieved by moving each transit, in turn, around the other nodes in the network while new added links are considered along with the removal of existing links. At each stage, any tested network proving to be cheaper than the previous one is accepted and the process repeated. The exact flow of this perturbation is very flexible. The general case is given by the following algorithm.

```
 1.  Select a set of transist, $J_1,J_2,\ldots,J_x$
 2.  Select a set of nodes, $i_1,i_2,\ldots,i_x$
 3.  Move each transit J to node (site) i
 4.  Reconnect core network as before
 5.  Reconnect each node to its nearest transit.
 6.  Select a different set of transits, $K_1,K_2,\ldots,K_y$
 7.  Connect each J and K
 8.  Select a different set of transits, $L_1,L_2,\ldots,L_z$
 9.  Disconnect each J and L
10.  Calculate cost of new network
11.  If there is an improvement in cost then 11. (a) Keep new
                                                     network
                                             11. (b) Goto 1
                                        else 11. (c) Discard
                                                     network
12.  If there are other choices of the L set then goto 8
13.  If there are other choices of the K set then goto 6
14.  If there are other choices of the i set then goto 2
15.  If there are other choices of the J set then goto 1
16.  Output Network
```

This algorithm will loop until there is no further improvement in cost to be achieved by any other local transformation. At each stage, the network undergoes a small change and the new cost is determined. The simplest version of the above algorithm is illustrated in Fig. 7 with x=y=z=1. Once this process is completed, the location of each transit is established permanently.

A somewhat different application of the concept of local perturbations applied to an initial solution is given by Rothfarb et al (1969).

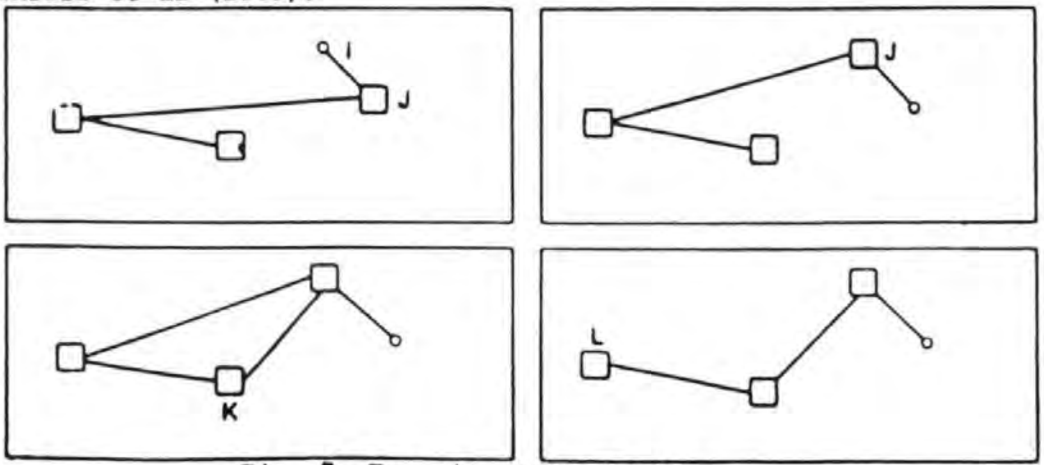


Fig. 7 Transit perturbation

3.5 Core network optimisation

At this point the number and location of the M transits has been determined. Each node is connected directly to its nearest transit and it only remains to determine the best way of connecting these M transits together.

With M transits there are $2^{(M(M-1)/2)-1}$ possible core networks. For 10 transits, there is approximately 10^{13}, far too many to examine individually. In addition, the appropriate routeing strategy has to be determined as well. The number of such feasible routeings depends on the particular topology of the core network under consideration. As an example, consider a full mesh on M transits with a DOR of 1. Each of the M(M-1)/2 pairs of nodes can either communicate directly or via any one of the M-2 other nodes in the network. Hence there are $[M-1]^{M(M-1)/2}$ possible routeing strategies for the mesh alone. For M=10, this is 10^{44}. Once again an exhaustive search can be ruled out as a possible method of solution and a more efficient technique is required to determine the core network. The method presented here determines both the topology and the routeing at the same time. Firstly, a simple method is given, similar in principle to the drop algorithm of Bahl and Tang (1972). This is then extended to give a much more powerful technique for determining the core network.

To begin with, each transit is connected to every other transit in a full mesh arrangement. Any two transits communicate along the direct link between them. A number of trial removals are then attempted. Each link, in turn, is removed and the traffic along that link is rerouted via each of the other transits in turn. Of all possible link removals and all possible rerouteings, one choice will lead to the greatest improvement in cost. This particular modification to the network is then made permanent. The link is removed and the traffic along that link rerouted accordingly. At each stage, the constraints of DOR and network security are checked and no link removal is allowed which would violate these conditions. This process continues, step by step, until the greatest improvement which could be achieved by a further link removal is negative. The optimisation is then complete. The logical flow of this technique is shown in Fig. 8. It can be seen that a strict decision is taken at each stage as to which path to follow and all other paths are discarded. As before, the problem still remains that a link could be removed at an early stage which would then prove to be of value in the final network. Once such a choice is made, however, the link cannot be replaced.

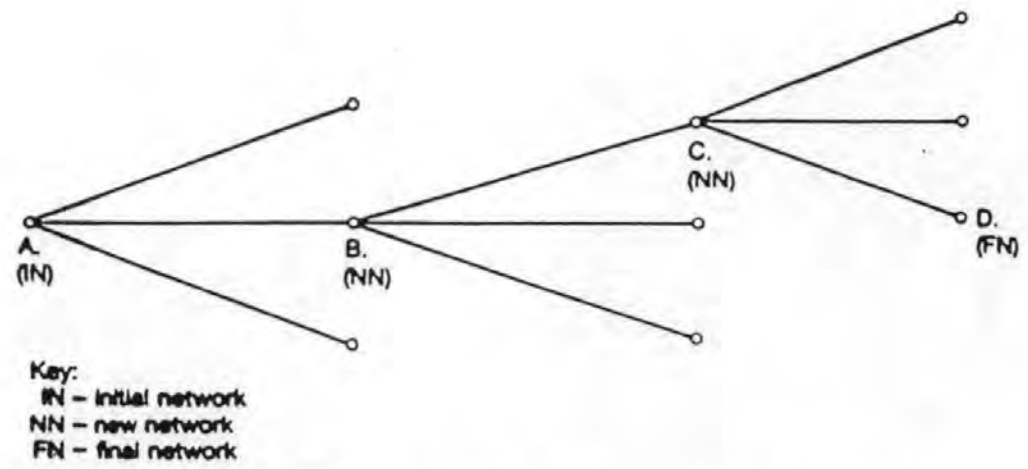


Fig. 8 Core network optimisation: simple method

This simple method can be improved upon by extending it to the look ahead process shown in Fig. 9. Instead of merely considering the removal of one link at a time, a sequence of link removals is considered up to S stages into the future, along with the accompanying rerouteings. The link which is eventually removed is that link whose path leads to the greatest improvement S stages ahead. The process then moves forward one step (B in Fig. 9) and looks ahead another S stages. Again the process continues until no improvement is possible.

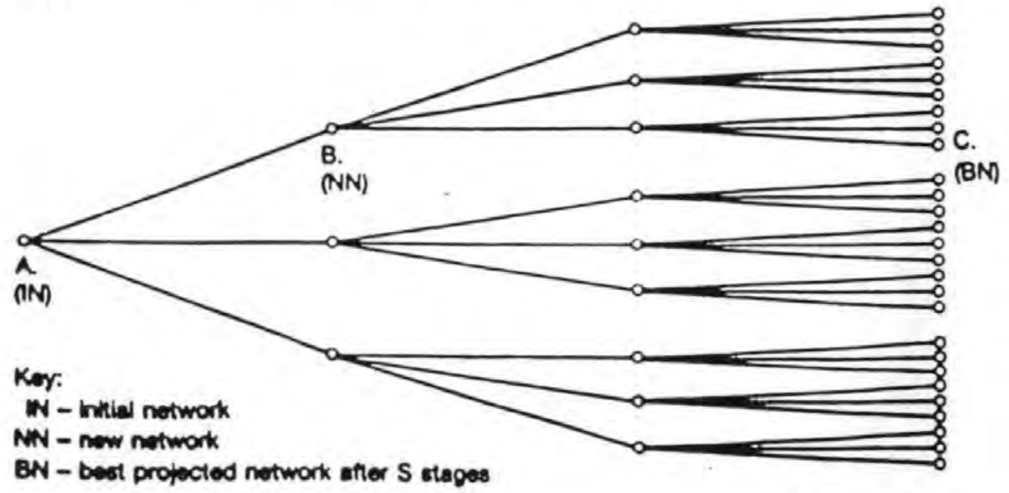


Fig. 9 Core network optimisation: improved method

Clearly, the accuracy of this method depends on the value of S and this accuracy has to be balanced against the time taken to complete the process. If time permitted, it is possible, in theory, to make S so large that all possible networks were considered.

Lin & Kernighan (1973) briefly discuss a form of look ahead technique as a possible extension to their Travelling Salesman algorithm.

3.6 Service network optimisation

Although the service network links can be either analogue or digital, results have shown that if all links are considered digital from the start, satisfactory networks are obtained. At this stage a simple comparison is made between the cost of a number of analogue circuits needed to carry the traffic for each of the n-M links with the cost of the equivalent digital circuits, the cheaper alternative being chosen in each case. As the traffic in the network under consideration grows and the trend towards cheaper digital links continues, any errors at this stage become negligible.

3.7 Combined operation and implementation

The algorithm has been presented in five sections since its flexibility and modular operation are extremely powerful in this form. The accuracy can be made increasingly good by allowing longer running times. Alternatively, remarkably good results can be obtained very quickly by minimising the relevant running times wherever possible. There are three main areas of flexibility.

(a) The value of q. Reduction is reasonably quick, particularly in the latter stages (since there are fewer distances to compare). The amount of work done by the star optimisation section, however, clearly depends on the number of replacement nodes it is acting upon. The greater the value of q, the more accurate the results and the longer the running time.

(b) The method of perturbation. Clearly, there are many ways in which transits can be moved around the network while links are added and removed. Algorithm 3.4.1 is only one example. The more extensive the perturbations, the less likely it will be that an optimum is missed but the longer the running time.

(c) The value of S. As outlined before, the solution will become more and more accurate as the number of look ahead stages increases but this will, of course, increase the running time.

The techniques presented in this paper, have been coded in Pascal in the five separate sections as they are given and set into operation on an IBM PC-XT personal computer. The results obtained are contained in the next solution.

4. NUMERICAL RESULTS

In order to examine the techniques in detail, an exhaustive search or full optimisation program was developed to run with identical network examples to those tested using the above methods, but on a large PRIME 9950 computer. This is a fast, powerful machine by comparison with the IBM PC-XT and is capable of making large numbers of calculations in a short time. Even so, exact optimisation methods of this type still take huge amounts of cpu time due to the number of possibilities to be tested. Although such a method is very slow, requiring run times in the order of days, or even weeks, the exhaustive search program provides exact reference networks which are known to be correct. It also gives a measure of the complexity of the problem to note that this program becomes impossible to run for networks of more than 30 or 40 nodes. Comparisons of results between the IBM model and the PRIME solutions are very good indeed. Of 23 networks tested simultaneously, the overall optimum (number of transits, $M^{\hat{}}$, as well as combination of $M^{\hat{}}$ transits) was obtained in all but one case (the third network shown in Table I). This was an artificial network deliberately constructed in a totally symmetric form with completely uniform traffic flow. Its purpose was to test the model to the extremes of its operation. Even in this instance, the difference in cost between the modelled and the exact solutions was less than 1%. If the number of transits was constrained to be artificially higher or lower than $M^{\hat{}}$, for the purposes of determining the optimum network with a specific number of transits, the results obtained were exact around 90% of the time with the error, in the remaining case, being bounded by 3% and averaging less than 1%. These should be considered excellent results for a program running so much faster than existing methods and only on a personal computer. It appears that only when particularly suboptimal numbers of transits are requested by the user, that the model begins to lose accuracy and even then, to within acceptable limits. A selection of six typical networks are shown in Table I.

The precise time taken by the model to obtain a solution is somewhat flexible (see section 3.7). Fig. 10 gives an example of the saving in time achieved by the reduction process and the assumption of a star at the early stages. It shows graphically the number of different networks which need to be considered, firstly by the full optimisation method and then by the method presented in this paper. A typical 100 node

Table 1

Results of comparisons between the PRIME and the IBM PC-XT

Network no.	No. of nodes	No. of transits	Cost of Solution Produced by PRIME	Cost of Solution Produced by IBM PC-XT	Difference (%)
1	19	1	1.997	1.997	0.0
		2	1.603	1.603	0.0
		3	1.591	1.591	0.0
		4	1.590*	1.590*	0.0
		5	1.668	1.700	0.1
2	50	1	1.532	1.532	0.0
		2	1.219	1.219	0.0
		3	1.199	1.199	0.0
		4	1.190*	1.190*	0.0
		5	1.203	1.203	0.0
		6	1.256	1.256	0.0
3	49	1	2.340	2.340	0.0
		2	2.254	2.254	0.0
		3	2.152	2.152	0.0
		4	2.093	2.093*	0.0
		5	2.079*	2.095	0.8
		6	2.146	2.155	0.4
4	18	1	1.512	1.512	0.0
		2	1.484*	1.484*	0.0
		3	1.509	1.509	0.0
		4	1.562	1.562	0.0
5	25	1	0.965	0.965	0.0
		2	0.952*	0.952*	0.0
		3	1.001	1.001	0.0
		4	1.090	1.122	2.9
6	21	1	1.935	1.935	0.0
		2	1.895	1.895	0.0
		3	1.891*	1.891*	0.0
		4	1.899	1.908	0.4

* best overall solution for each network by each method.

network with an optimum of 4 transits requires the following approximate times for each section.

1. Reduction	: From n=100 to q=30	: 10 minutes
2. Star Optimisation	: From q=30 to M^ =4	: 40 minutes
3. Transit Perturbation	: From M^ =4 to new M^=4	: 120 minutes
4. Core Optimisation	: S=1	: 5 seconds
	: S=2	: 5 minutes
	: S=3	: 60 minutes
5. Service Optimisation		: 2 seconds

These timings are the approximate average values for each section, measured using an IBM PC-XT personal computer, a comparatively slow machine. If the programs are run on the PRIME, which is about 20 times fast, it can be seen that, in about 10 minutes, it would be possible to obtain an approximation to a solution that was otherwise impossible!

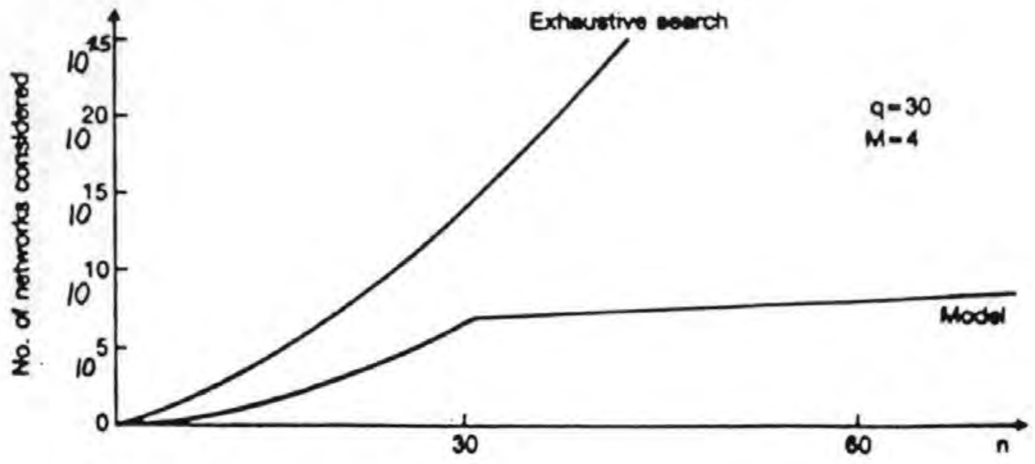


Fig. 10 Comparison of Computational Complexity

5. CONCLUSIONS

This is a new and fundamentally different method of optimising the structure of large switching networks. The results obtained to date are extremely encouraging and it appears that the combined objectives of decreasing the computational work whilst maintaining an acceptable level of accuracy have been achieved. It soon becomes impossible to test the model empirically as the number of nodes increases since exact methods require excessive amounts of processing time, even on the largest machines. Work is underway at present to determine the limitations of these techniques via a theoretical approach to the analysis. In addition, the extension of the model to more than two levels seems reasonable from preliminary studies.

Since no assumptions have been made that are based solely on telecommunication networks, the application of these ideas, both to other types of network problem and combinatorial problems in general, seems possible.

ACKNOWLEDGEMENT

The results of this work will eventually form a software package to be used by British Telecom CFM. We gratefully acknowledge British Telecom plc. for their sponsorship of this work and their kind permission to present this paper.

8. REFERENCES

Bahl, L.R. and Tang, D.T., (1972) "Optimisation of Concentrator Locations in Teleprocessing Networks". Symposium on Computer-Communications Networks and Teletraffic, Polytechnic Institute of Brooklyn, April 4-6.

Chandy, K.M. and Russell, R.A., (1972) "The Design of Multipoint Linkages in a Teleprocessing Tree Network". *IEEE Trans. on Computers,* C-21, no. 10, 1062-1066.

Gunnarsson, R. and Nivert, K., (1984) "Optimum Grade of Service in Telecommunications Networks". *Telecommunication Jnl.,* 51, 323-326.

Hansler, E., McAuliffe, G.K. and Wilkov, R.S., (1972) "Optimising the Reliability in Centralised Computer Networks". *IEEE Trans. on Communications,* COM-20, no. 3, 640-644.

Hoshi, M., (1985) "Local Network Area Size Optimisation". *IEEE Trans. on Communications,* COM-33, No. 3, 199-202.

Karnaugh, M., (1976) "A New Class of Algorithms for Multipoint Network Optimisation". *IEEE Trans. on Communications,* COM-24, No. 5, 500-505.

Kershenbaum, A. and Chou, W., (1974) "A Unified Algorithm for Designing Multidrop Teleprocessing Networks". *IEEE Trans. on Communications, COM-22,* no. 11, 1762-1772.

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G. and Shmoys, D.B., (1985) "The Travelling Salesman Problem". Wiley Interscience Series in Discrete Mathematics.

Lin, S. and Kernighan, B.W., (1973) "An Effective Heuristic Algorithm for the Travelling Salesman Problem". *Oper. Res.*, **21**, 498-516.

Matsui, S., (1978) "A Network Optimisation by Mixed Integer Programming". Nippon Univac Kaisha Ltd., Tokyo, Japan.

Mirzaian, A., (1985) "Lagrangian Relaxation for the Star-Star Concentrator Location Problem: Approximation Algorithm and Bounds". Networks, 15, 1-20.

Rothfarb, B., Frank, H., Rosenbaum, D.M., Steiglitz, K. and Kleitman, D.J., (1970) "Optimal Design of Offshore Natural-Gas Pipeline Systems". *Oper. Res.*, **18**, 992-1020.

Tang, D.T., Woo, L.S. and Bahl, L.R., (1978) "Optimisation of Teleprocessing Networks with Concetrators and Multiconnected Terminals". *IEEE Trans. on Computers*, C-27, no. 7, 594-604.

COMMUNICATION NETWORK OPTIMISATION: PAST, PRESENT AND FUTURE.

V.M. Grout and P.W. Sanders,
Department of Communication Engineering,
Faculty of Technology,
Plymouth Polytechnic,
Drake Circus, PLYMOUTH,
Devon, PL4 8AA, UK.

January 1988.

ABSTRACT.

This paper surveys the major facets of the branch of combinatorial optimisation known as communication network optimisation. The basic principles are investigated briefly, followed by a discussion of location, connection and traffic routeing problems in turn. In each case the relevant existing algorithms are listed together with some suggestions concerning the likely areas of investigation in future years.

1. INTRODUCTION.

It is common for communication networks to make use of the most modern equipment available. Such equipment is, by definition, expensive and it clearly becomes important to use it in the most efficient way possible. It is this pursuit of perfection, in its various forms, which comprise the field of communication network optimisation.

There are a number of individual problems to be dealt with in the design of any particular network. Each may or may not be appropriate in any given situation. This paper considers the more important of these by presenting the major pieces of work which have been carried out in each case. Before this can be achieved, however, some basic ground rules are necessary.

1.1. Objectives.

Any variable system can only be optimised with respect to a single objective function although this may be a non-trivial combination of a number of components. The optimisation discussed in this paper will be mainly carried out with respect to minimising the cost of the resultant network. In general, a number of constraints will be stipulated and the network of minimum cost, satisfying these constraints, will be sought.

There are alternative forms of optimisation, however. If a predetermined amount of facilities are available then the cost can be considered as one of the fixed constraints and different parameters such as reliability (Hansler et al. [1972], Van Slyke & Frank [1972] and Wilkov [1972]), grade of service (Gunnarsson & Nivert [1984]), congestion (Davies [1972]) or traffic flow can be maximised or minimised as appropriate.

1.2. Simple Principles of Communication Networks.

The purpose of any network is to connect together a number of sources and sinks of communications traffic, known as <u>nodes</u>, distributed within a given region. The nodes are situated in such a way as to serve the underlying traffic distribution of the region. If the locations of the nodes are known then so is the amount of traffic flowing in each direction between each node pair. The number of nodes is commonly represented by n.

A solution to a network problem will consist of a system of <u>links</u> and <u>switches</u> (sometimes known as <u>tandems</u>), collecting and distributing the traffic around the network in the required manner. Both the topological structure of this network and the way in which the traffic flows around it are subjects for optimisation. There is clearly a strong relationship between these networks and the mathematical subject of graph theory (Harary [1971] and Wilson [1972]). This relationship is investigated in Cattermole [1975] and Grout [1988].

The cost of any network will be the sum of the individual costs of its component links, nodes and switches. The cost of a link, node or switch is dictated by its size which, in turn, depends on the amount of traffic it has to deal with (and in the case of a link, its length). For these purposes, approximations such as the Erlang B formula are generally used (Hills [1979]). It is the purpose of this paper to survey the principal techniques for optimisation available and not to deal, in any great detail, with the mathematical technicalities. For such a treatment, see Meister et al. [1972] or Grout [1988].

2. LOCATION OF FACILITIES.

The effective placement of nodes and switches will have a considerable influence upon the cost of the resultant network. Ideally, each should be situated in the location where it most efficiently collects (and distributes) the traffic which it has to deal with. This section deals with nodes and switches in turn.

2.1. Traffic Server Locations.

For any given region there will be a certain distribution of traffic generated and received. The nodes should be placed in such a way as to minimise the cost of serving this distribution. There are two problems to be dealt with: that of determining the number of nodes (and consequently the area covered by each node) and the location of each node within its respective area.

One of the first real attempts to deal with these problems is given in Rapp [1962]. It is noted that the cost of a node within a single traffic area is a function $F(x,y,D)$ of its x and y coordinates and the traffic distribution D. The optimum location of the node is then given by $\partial F/\partial x = \partial F/\partial y = 0$. The size of the node should then be calculated merely to be large enough to deal with the traffic in that area. The traffic distribution is typically represented by a numerical grid (Rapp [1962], Enriquez de Salamanca & Zulueta [1971] and Wadhwa [1979]) with larger values representing areas of greater traffic density. This situation is shown in Fig. 1.

In practice the node placement can be achieved with reasonable accuracy (bearing in mind that traffic density figures are likely to be imprecise anyway) by balancing moments or, even more simply, by ensuring that the traffic totals on either side of an imaginary straight line drawn through the node are equal. Similar work has been carried out by Okazaki [1982].

The situation is not, in truth, as simple as this. The ultimate objective in network design is to minimise the total cost. Placing the node in such a way as to minimise the cost of the subscriber connections in that group may cause the links between the node and its parent tandem(s) to be longer than necessary. If the cost of these links is considered, the effect is to pull the node away from the centre of the group (see Fig. 2). Taking this into account, the problem of node placement is more difficult and may be a productive area of investigation for the future.

2.2. Switching Centre Locations.

If the positions of the n nodes are established it then becomes necessary to determine the optimum location of the M switches in the higher level network as well as the best value of M. A number of constraints are possible at this stage, depending upon the form of the solution required.

Hoshi [1985] addresses the problem of finding the optimum size of each tandem group. Clearly the mean size of each group is inversely proportional to the number of tandems. A number of somewhat simplistic assumptions are made in this paper but it does provide an explicit method for determining M.

The general problem of determining the best choice of tandems has been considered by Bahl & Tang [1972], Boorstyn & Frank [1977], Tang et al. [1978], Hoc [1982], Mirzaian [1985] and Grout et al. [1986, 1987] among many others. The approaches are many and varied. Some algorithms only work for networks of a specific type whereas others tend to make impractical assumptions such as knowing the cost of variable amounts of equipment in advance or having it proportional to a simple formula (see Grout [1988] for a fuller discussion of the shortcomings of some existing methods).

The general problem is extremely difficult. To know the cost of a tandem requires that its size be known also. This will depend on the closeness of the other tandems which, in turn, implies prior knowledge of the tandem locations, the very thing required at the beginning. The only escape is to consider each selection of tandems on its merits, one at a time. To examine every tandem set in this way, however, would take too long and, in practice, heuristic techniques such as the Double Drop method (BP [1984]) or that of Grout et al. [1986,1987] are used instead. These methods do not guarantee the exact optimum solution but are found, in practice, to provide approximations of acceptable accuracy. The Double Drop method begins with all nodes established as switches and systematically removes them until the required set is reached. The method of Grout et al. employs a reduction stage to simplify the problem followed by a constrained technique for tandem selection.

The most profitable direction for research in the future would be into ways of improving these practical heuristics, both in terms of speed of execution and accuracy of results. The continually improved computational power becoming available (Sumner [1986]) would appear to ensure the former to some extent.

3. NETWORK CONNECTION TOPOLOGIES.

With the position of the n nodes and M switches established, the problem becomes one of connection rather than location. There are at least as many algorithms for the connection problem as for the location problem (indeed some methods deal with both simultaneously), each varying according to the particular application it is intended to deal with. Firstly, a special type of network is considered, followed by the more general case.

3.1. Centralised Networks.

These networks are those of the form discussed by Chandy & Russell [1972] and correspond to systems where communication is only required between a number of terminals and a central computer or data base (see Fig. 3). A number of algorithms exist for their solution. Most are based, in some way or another, on tree structures (Kruskal [1956] and Prim [1957]). The minimal spanning tree (MST) of a set of nodes can be computed easily (Shamos [1978]). If the MST were always suitable the problem would be solved. However practical requirements of link capacity and reliability usually imply a series of constraints upon the final solution which complicate the situation.

Exhaustive search techniques are not desirable if the number of nodes is large. Chandy and Russell [1972] present a method for determining the optimum network based on the branch and bound work of Little et al. [1963]. This is still, in effect, merely a structured exhaustive search and ultimately heuristics are necessary.

Both the algorithms of Kruskal [1956] and Prim [1957] can be adapted to give constrained heuristic solutions although superior results are obtained via the algorithm of Esau and Williams [1966] in which a 'trade-off' function is maximised in order to determine the best way to reconnect the network at each stage. All three, however, can be shown to be variants of a single method (Kershenbaum & Chou [1974]). Further heuristics are offered by Frank et al. [1971] and Elias & Ferguson [1974].

For networks of moderate size, the improved techniques of Karnaugh [1976] and Kershenbaum et al. [1980] may be suitable. These are more accurate algorithms which will take longer to execute than the simple method above. Karnaugh's method, for example, makes repeated calls to a modified Esau Williams algorithm and produces results which are 2-3% better than those given otherwise.

The work carried out by Tang et al. [1978] and Hoc [1982] also deals with the choice of links in addition to the selection of concentrators to which the papers are more earnestly directed.

3.2. Generalised Networks.

This section deals with the more general case of section 3.1. This involves a number of nodes and switches in which traffic may flow from any node to any other node in the network. The situation is clearly far more complicated and, not surprisingly, fewer real algorithms exist for its solution.

In contrast, however, there is no shortage of literature concerning the problems involved. Gallego [1971], Frank & Chou [1972], Goldstein [1973], Engvall [1979] and Greenhop & Campbell [1984] all address themselves to the matter in various forms. The American ARPA network, in particular, has been studied in great detail (Frank et al. [1970], Cole [1972], Frank et al. [1972] and Shwartz [1977]).

Whilst providing powerful and useful insights into the behavior of operational communication networks, much of the work carried out to date does not allow genuine optimisation to be performed on a general network problem. It is certainly difficult, and probably impossible, to represent the complexities of a typical network problem by a mathematical model of a suitable form to allow conventional problem solving techniques to operate. For example, while linear programming problems are relatively easy to solve (Karmarkar [1984]), to express a network optimisation problem in such a form requires significant simplification in most areas. The results obtained via such an approach should thus be treated with extreme caution.

The natural alternative to modelling techniques, the systematic generation and evaluation of all possible solutions, is, whilst accurate, far too complex a task to consider for all but very small problems. The number of possible choices of M tandems from n nodes, for example, is ${}^{n}C_{M}$ so that the total number of valid sets is $\Sigma_{M=1..n}({}^{n}C_{M})$ and this takes no account of link placement or traffic routeing.

As before, heuristics are clearly needed to deal with large problems. The Double Drop method (BP [1984]) and the method of Grout et al. [1986, 1987] mentioned in section 2.2 for determining tandem locations also provide methods of approximating the optimum link structure, the method of Grout et al. being a generalisation of the

Double Drop method. As before, it may be expected that the future will yield further heuristics for improving the accuracy of optimisation of the larger network problems. At present such techniques are extremely rare.

4. TRAFFIC ROUTEING STRATEGIES.

Suppose now that the topological (i.e. physical) structure of the solution network is determined. The locations of the nodes and switches are established and the appropriate links have been added between them. The result can be regarded as a graph on the n nodes. The solution is not complete, however. For a given node pair (i, j), there will be, in general, a number of paths from i to j. The problem of finding the best path for each node pair (remembering that each will interact) forms the basis of all traffic routeing algorithms. The problem, however, can manifest itself in a number of forms.

4.1. Shortest Path Problems.

The simplest form of routeing problem concerns finding the shortest path between i and j for each (i, j) in the network. It is simpler to merely consider two nodes at a time and to repeat the process as necessary. Suppose then that the shortest path between nodes u and v is required and that distances between all pairs of nodes in the network are known.

One of the first attempts at solving this problem is by Ford [1956] although some work was done earlier by Heller [1953]. The method given produces the shortest path in all cases but is not particularly computationally efficient. Dijkstra [1959] suggests an alternative which is probably the simplest of all shortest path algorithms.

Dijkstra's algorithm operates by including each node in turn into a set of nodes whose distance from u is known. Initially only u is in this set. At each subsequent stage the closest node to a node already in the set is chosen and its distance from u calculated as the minimum of its distance from w plus the distance of w from u for all w already in the set. The process continues until v becomes included in the set in this way. Dijkstra's algorithm is an improvement over Ford's because only a relatively small amount of work has to be done at each stage both in terms of computation and storage requirements.

Golden [1976] notes that an algorithm due to Bellman [1958] is more computationally efficient than Dijkstra's for sparse graphs with Euclidean distances and Golden & Ball [1978] suggest an improved version of Dijkstra's algorithm which also performs better than the original for similar examples.

A comparable problem, that of finding the shortest path through a number of nodes, is considered by Verblunsky [1951] and Beardwood et al. [1959]. The problem then increases considerably in complexity, however, and more resembles the famous travelling salesman problem (Lawler et al. [1985]).

4.2. Maximum Flow Problems.

Another way in which a traffic routeing problem can be expressed is in terms of finding the maximum traffic flow in the network. Assuming that the network is currently designed to carry the required amount of traffic, an effective choice of routeing may lead to a surplus capacity in some areas which can be utilised sometime into the future. It may be of great importance then to route all traffic around the network so that the maximum flow is possible.

To date, the major part of the results achieved with respect to maximum flow problems has been theoretical although in some instances the progression to the practical case is not extremely hard. It is assumed that all links within the network have known and fixed capacities. For any two nodes, u and v, within the network, the problem of finding the maximum possible flow from from u to v is a well solved one.

Menger [1927] showed that the maximum number of separate paths between u and v is equal to the minimum number of links needed to disconnect u from v completely (Harary [1971]). This result can be extended (Wilson [1972]) to show that the maximum flow possible between u and v is equal to the minimum sum of the capacities of such a set of links. Based upon this observation, Ford & Fulkerson [1956] propose an algorithm for finding such a flow based on a sequence of gradual improvements applied to a starting solution. The problem of calculating the magnitude of such a maximum flow with practical restrictions upon the length of each path, however, is known to be hard (Itai et al. [1982]). A heuristic procedure for this purpose is proposed by Ronen & Peri [1984].

The real problem, of course, is far more involved than this. Instead of traffic flowing between two nodes in isolation, all pairs of nodes in the network may need to communicate. Traffic between one pair may share links and switches with that between another pair (the capacity of the switches is not even considered by the simple algorithms although it can be incorporated into the capacity of the links with care). This problem has barely been approached in the literature on network optimisation and may well provide fruit if some work was to be directed at it.

4.3. General Routeing Problems.

The original and most common problem, however, is that of minimising the cost of the network. This involves the problem of routeing the traffic, not so as to minimise distance or maximise flow arbitrarily, but to find the most economical strategy for whatever form of cost is being used. Not surprisingly, the problem is far harder than the previous two and not particularly well solved in the practical case although there are some good examples of theoretical work available (Frank & Chou [1971], Gallager [1977], Yu et al. [1984], Esfahanian & Hakimi [1985] and Mars & Narendra [1987]). Many of these, however, attempt to restrict congestion etc. as opposed to minmising cost. Mulvey [1978] deals with a simplification of the minimum cost flow problem using the simplex linear programming method.

The real problem, however, is so much more difficult due, as much as any thing to the number of restrictions which may in force as well as the sheer number of routeings that are possible. Even if traffic is only permitted to pass via a maximum of one other node or tandem, there are of the order of n^{n^2} possible overall constructions (see Grout [1988]).

Again, practical techniques such as the double drop method (BP [1984]) and the method of Grout et al. [1986,1987] can achieve heuristic results to such problems but their accuracy in this case is not entirely proven. This is possibly one area of optimisation, if any, which urgently needs a considerable improvement in the availability of practical results and algorithms.

5. CONCLUSIONS.

Clearly there is no shortage of literature available concerning network optimisation in all its forms. In particular, the simplifications of the general case appear to be well solved. The more general problem, however, is still proving difficult and may well remain so for some time (see Greenhop & Campbell [1984] or Grout [1988] for further discussion).

6. REFERENCES.

Bahl, L.R. & Tang, D.T., 'Optimization of Concentrator Locations in Teleprocessing Networks', Proc. of the Symposium on Computer Communications Networks and Teletraffic', Polytechnic Institute of Brooklyn, April 4-6 1972, pp355-362.

Beardwood, J., Halton, J.H. & Hammersley, J.M., 'The Shortest Path Through Many Points', Proc. of the Cambridge Philosophical Society, Vol. 55, 1959, pp299-327.

Bellman, R., 'On a Routing Problem', Quart. Applied Mathematics, Vol. 16, 1958, pp87-90.

Boorstyn, R.R. & Frank, H., 'Large-Scale Network Topological Optimization', IEEE Trans. on Communications, Vol. COM-25, No. 1, January 1977, pp29-47.

BP, 'Group (UK) Telecommunications Network Design Study', BP Report No. TEL 00184, February 1984.

Cattermole, K.W., 'Graph Theory and the Telecommunications Network', IMA Bulletin, May 1975, pp94-107.

Chandy, K.M. & Russell, R.A., 'The Design of Multipoint Linkages in a Teleprocessing Tree Network', IEEE Trans. on Computers, Vol C-21, No. 10, October 1972, pp1062-1066.

Cole, G.D., 'Performance Measurements on the ARPA Computer Network', IEEE Trans. on Communications, Vol. COM-20, No. 3, June 1972, pp630-636.

Dijkstra, E.W., 'Two Problems in Connexion with Graphs', Numerische Mathematik, Vol. 1, 1959, pp269-271.

Davies, D.W., 'The Control of Congestion in Packet-Switching Networks', IEEE Trans. on Communications, Vol. COM-20, No. 3, June 1972, pp546-550.

Elias, D. & Ferguson, M.J., Topological Design of Teleprocessing Networks', IEEE Trans. on Communications, Vol. COM-22, No. 11, November 1974, pp1753-1762.

Engvall, L., 'International Network Optimization Methods', Telecommunication Journal, Vol. 46, XI, 1979, pp689-695.

Enriquez de Salamanca, M. & and Zulueta, J., 'Computer Aids to Network Planning: Placement of Telephone Exchanges in Urban Areas', Electrical Communication, Vol. 46, No. 3, 1971, pp196-201.

Esau, L.R. & Williams, K.C., 'On Teleprocessing System Design: Part II: A Method for Approximating the Optimal Network', IBM Systems Journal, Vol. 5, No. 3, 1966, pp142-147.

Esfahanian, A-H. & Hakimi, S.L., 'Fault Tolerant Routing in DeBruijn Communication Networks', IEEE Trans. on Computers, Vol. C-34, No. 9, September 1985, pp777-788.

Ford, L.R., 'Network Flow Theory', Rand Corporation Paper P-923, 1956.

Ford, L.R. & Fulkerson, D.R., 'Maximal Flow Through a Network', Canadian Journal of Mathematics, Vol. 8, 1956, pp399-404.

Frank, H., Frisch, I.T. & Chow, W., 'Topological Considerations in the Design of the ARPA Computer Network', Proc. of the AFIPS Spring Joint Computer Conference, 1970, pp581-587.

Frank, H., Frisch, I.T., Van Slyke, R. & Chou, W.S., 'Optimal Design of Centralized Computer Networks', Networks, Vol. 1, 1971, pp43-57.

Frank, H. & Chou, W., 'Routing in Computer Networks', Networks, Vol. 1, 1971, pp99-112.

Frank, H. & Chou, W., 'Topological Optimization of Computer Networks', Proc. of the IEEE, Vol. 60, November 1972, pp1385-1397.

Frank, H., Kahn, R.E. & Kleinrock, L., 'Computer Communication Network Design - Experience with Theory and Practice', Proc. of the AFIPS Spring Joint Computer Conference, 1972, pp255-270.

Gallager, R.G., 'A Minimum Delay Routing Algorithm Using Distributed Computation', IEEE Trans. on Communications, Vol. COM-25, No. 1, January 1977, pp73-85.

Gallego, P.A.C., 'National Trunking Network Optimization', Electrical Communication, Vol. 46, No. 3, 1971, pp202-206.

Golden, B., 'Shortest Path Algorithms: A Comparison', Operations Research, Vol. 24, No. 6, November-December 1976, pp1164-1168.

Golden, B.L. & Ball, M., 'Shortest Paths with Euclidean Distances: An Explanatory Model', Networks, Vol. 8, 1978, pp297-314.

Goldstein, M.C., 'Design of Long Distance Telecommunication Networks - The Telpak Problem', IEEE Trans. on Circuit Theory, Vol. CT-20, No. 3, May 1973, pp186-192.

Greenhop, D. & Campbell, R., 'COST 201 - A Procedure for the Optimisation of Telecommunication Networks', British Telecommunications Engineering, Vol. 3, April 1984, pp47-58.

Grout, V.M., 'Optimisation Techniques for Telecommunication Networks', Ph.D. Thesis, Department of Communication Engineering, Faculty of Technology, Plymouth Polytechnic, UK, 1988.

Grout, V.M., Sanders, P.W. & Stockel, C.T., 'Large Scale Telecommunication Network Optimisation', Proc. of the IMA Conference on Simulation and Optimisation of Large Systems, University of Reading, September 23-25 1986.

Grout, V.M., Sanders, P.W. & Stockel, C.T., 'Practical Approach to the Optimisation of Large-Scale Switching Networks', Computer Communications, Vol. 10, No. 1, February 1987, pp30-38.

Gunnarsson, R. & Nivert, K., 'Optimum Grade of Service in Telecommunication Networks', Telecommunication Journal, Vol. 51, VII, 1984, pp323-326.

Hansler, E., McAuliffe, G.K. & Wilkov, R.S., 'Optimizing the Reliability in Centralized Computer Networks', IEEE Trans. on Communications, Vol COM-20, No. 3, June 1972, pp640-644.

Harary, F., 'Graph Theory', Addison Wesley, 1971.

Heller, I., 'On the Problem of Shortest Paths Between Points, I and II', Bull. of the American Mathematical Society, Vol. 59, 1953, pp551-552.

Hills, M.T., 'Telecommunication Switching Principles', Allen & Unwin, 1979.

Hoc, H.H., 'Topological Optimization of Networks: A Nonlinear Mixed Integer Model Employing Generalized Benders Decomposition', IEEE Trans. on Automatic Control, Vol. AC-27, No. 1, February 1982, pp164-169.

Hoshi, M., 'Local Network Area Size Optimization', IEEE Trans. on Communications, Vol. COM-33, No. 3, March 1985, pp199-202.

Itai, A., Peri, Y. & Shiloah, Y., 'The Complexity of Finding Maximum Disjoint Paths with Length Constraints', Networks, Vol. 12, 1982, pp277-286.

Karmarkar, N., 'A New Polynomial Time Algorithm for Linear Programming', Combinatorica, Vol. 4, 1984, pp373-395.

Karnaugh, M., 'A New Class of Algorithms for Multipoint Network Optimization', IEEE Trans. on Communications, Vol. COM-24, No. 5, May 1976, pp500-505.

Kershenbaum, A., Boorstyn, R. & Oppenheim, R., 'Second Order Greedy Algorithms for Centralized Teleprocessing Network Design', IEEE Trans. on Communications, Vol. COM-28, No. 10, October 1980, pp1835-1838.

Kershenbaum, A. & Chou, W., 'A Unified Algorithm for Designing Multidrop Teleprocessing Networks', IEEE Trans. on Communications', Vol. COM-22, No. 11, November 1974, pp1762-1771.

Kruskal, J.B., 'On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem', Proc. of the American Mathematics Society, Vol. 7, 1956, pp48-50.

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G. & Shmoys, D.B., 'The Traveling Salesman Problem', Wiley, 1985.

Little, J.C.D., Murty, K., Sweeney, D.W. & Karel, C., 'An Algorithm for the Traveling Salesman Problem', Operations Research, Vol. 11, November-December 1963, pp972-989.

Mars, P. & Narendra, K.S., 'Routing, Flow Control and Learning Algorithms', Proc. of the First IEE National Conference on UK Telecommunications Networks - Present and Future, June 2-3 1987, pp78-83.

Meister, B., Muller, H.R. & Rudin, H.R., 'On the Optimization of Message-Switching Networks', IEEE Trans. on Communications, Vol. COM-20, No. 1, February 1972, pp8-14.

Menger, K., 'Zur Allgemeinen Kurventheorie', Fund. Math., Vol. 10, 1927, pp96-115.

Mirzaian, A., 'Lagrangian Relaxation for the Star-Star Concentrator Location Problem: Approximation Algorithm and Bounds', Networks, Vol. 15, 1985, pp1-20.

Mulvey, J.M., 'Testing of a Large Scale Network Optimization Program', Mathematical Programming, Vol. 15, 1978, pp291-314.

Okazaki, H., 'Planning Digital Exchange Locations and Boundaries in Urban Area', NEC Research and Development, No. 66, July 1982, pp7-15.

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

Ronen, D. & Peri, Y., 'Heuristics for Finding a Maximum Number of Disjoint Bounded Paths', Networks, Vol. 14, 1984, pp531-544.

Rapp, Y., 'Planning of Exchange Locations and Boundaries in Multi-Exchange Networks', Ericsson Technics, No. 2, 1962, pp94-113.

Shamos, M.I., 'Computational Geometry', Ph.D. Thesis, Yale University, USA, 1978.

Shwartz, M., 'Computer Communication Network Design and Analysis', Prentice Hall, 1977.

Sumner, F.H., 'The Provision of High Performance Computing', Proc. of the IMA Conference on Simulation and Optimisation of Large Systems', University of Reading, September 23-25 1986.

Tang, D.T., Woo, L.S. & Bahl, L.R., 'Optimization of Teleprocessing Networks with Concentrators and Multiconnected Terminals', IEEE Trans. on Computers, Vol. C-27, No. 7, July 1978, pp594-604.

Van Slyke, R. & Frank, H., 'Network Reliability Analysis: Part 1', Networks, Vol. 1, 1972, pp279-290.

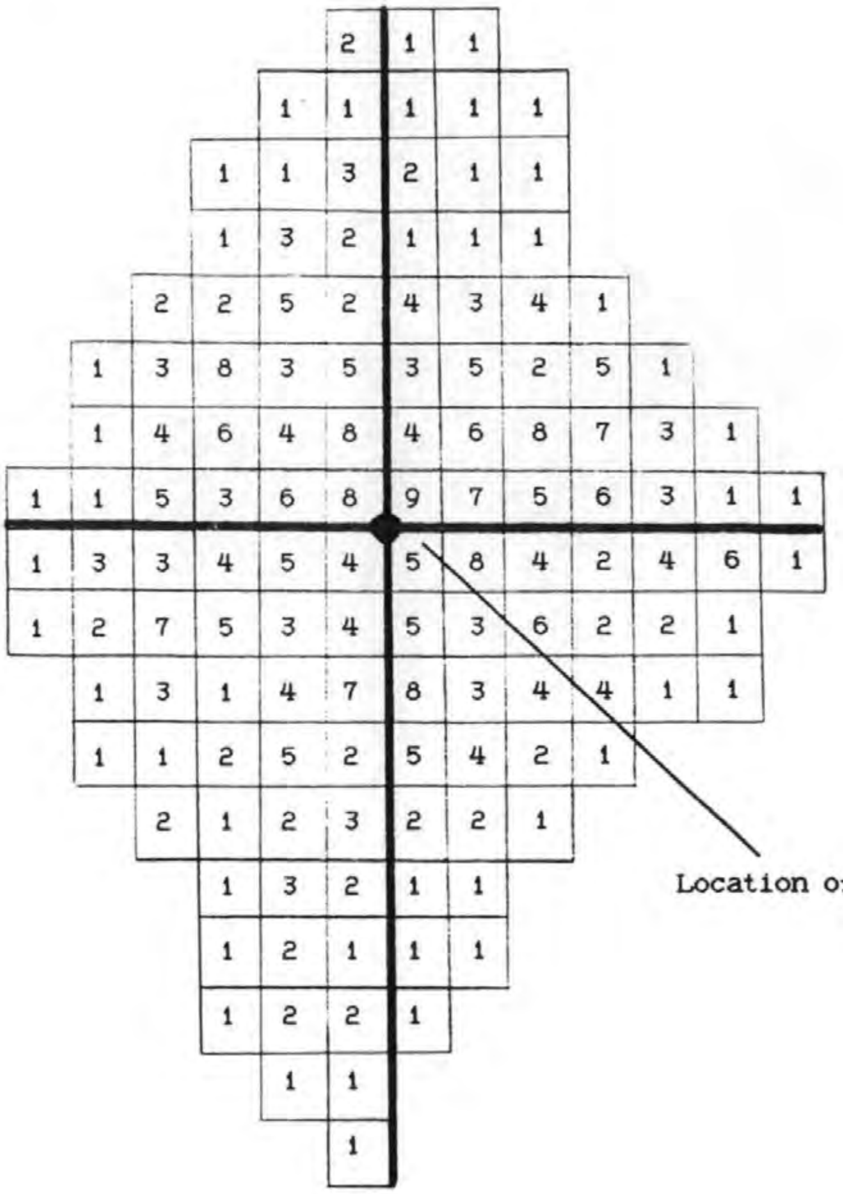
Verblunsky, S., 'On the Shortest Path Through a Number of Points', Proc. of the American Mathematical Society, Vol. 2, No. 6, pp904-913.

Wadhwa, T.R., 'Local Network Optimization', Telecommunications, Vol. 29-1, June 1979, pp15-19.

Wilkov, R.S., 'Analysis and Design of Reliable Computer Networks', IEEE Trans. on Communications, Vol. COM-20, No. 3, June 1972, pp660-678.

Wilson, R.J., 'Introduction to Graph Theory', Longman, 1972.

Yu, W., Majithia, J.C. & Wong, J.W., 'Distributed Routing Algorithm for a Circuit Switched Packet Switching Network', IEEE Proceedings, Vol. 131, Pt. F, No. 7, December 1984, pp751-760.



					2	1	1					
				1	1	1	1	1				
			1	1	3	2	1	1				
			1	3	2	1	1	1				
		2	2	5	2	4	3	4	1			
	1	3	8	3	5	3	5	2	5	1		
	1	4	6	4	8	4	6	8	7	3	1	
1	1	5	3	6	8	9	7	5	6	3	1	1
1	3	3	4	5	4	5	8	4	2	4	6	1
1	2	7	5	3	4	5	3	6	2	2	1	
	1	3	1	4	7	8	3	4	4	1	1	
	1	1	2	5	2	5	4	2	1			
		2	1	2	3	2	2	1				
			1	3	2	1	1					
			1	2	1	1	1					
			1	2	2	1						
				1	1							
					1							

Figure 1. Subscriber Density.

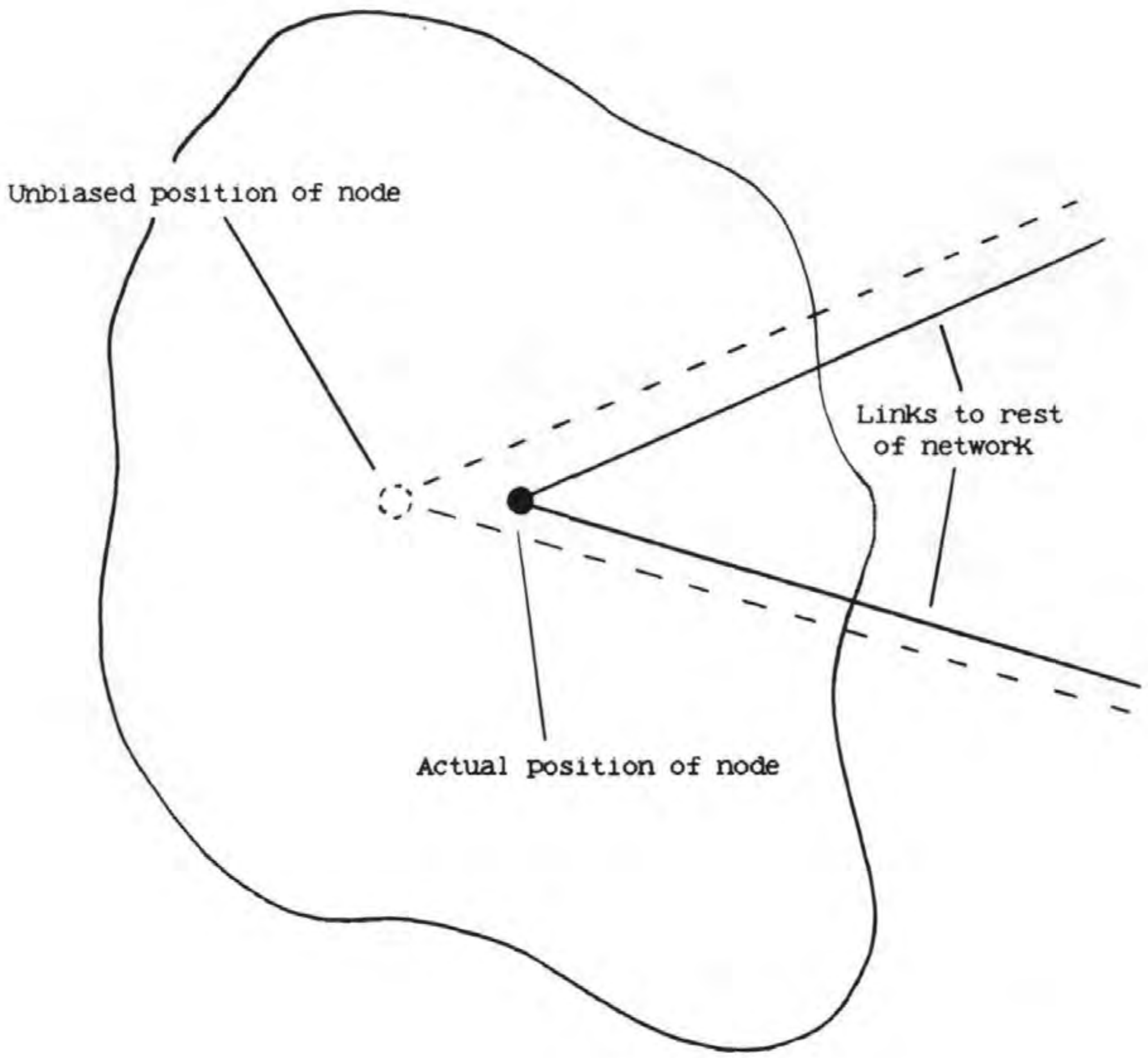


Figure 2. Node Location.

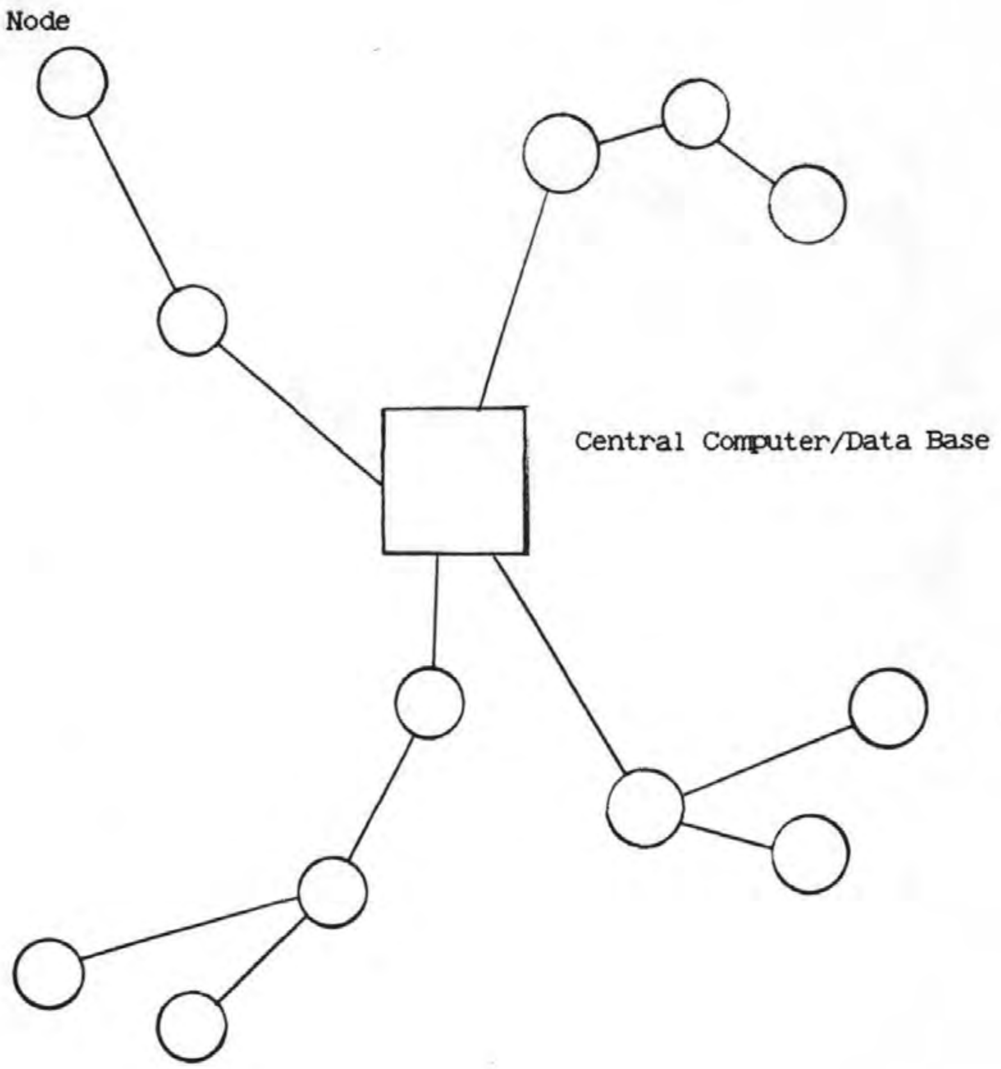


Figure 3. Centralised Network.