TRAFFIC ASSIGNMENT AND NETWORK ANALYSIS

A Thesis submitted for the degree of)

DOCTOR OF PHILOSOPHY

at

The University of London

by

Nather Zeki Simhairi

Royal Holloway and Bedford New College

1987
ACKNOWLEDGEMENT

The work described in this thesis was carried out under the supervision of Dr. A. Sheer in the department of mathematics, Royal Holloway and Bedford New College, between May 1982 and September 1985.

I wish to thank both Dr. A. Sheer for his encouragement, advice and guidance throughout this work and my parents for their continuous care and support.

I dedicate this work to my loving wife Mary.
CONTENTS

Title-Page 1
Acknowledgement 2
Abstract 5
Chapter 1 6
  Introduction 6
Chapter 2 (Transportation Networks) 10
  2.1 General Description 11
    2.1.1 Demand 11
    2.1.2 Supply 12
  2.2 The Traffic Assignment Model 13
    2.2.1 System-Optimal Flow 15
    2.2.2 User-Equilibrium Flow 15
  2.3 Traffic Assignment Techniques 16
Chapter 3 (Traffic Assignment) 21
  3.1 Introduction 21
  3.2 Finding Plausible Used Routes 24
  3.3 An Algorithm For Finding The User-Equilibrium Assignment 24
  3.4 Paradoxical Behaviour Of The Links 33
Chapter 4 (Linear Sensitivity Analysis To Detect Paradoxical And Cost Sensitive Links) 35
  4.1 Introduction 35
  4.2 Notation And Assumptions 37
  4.3 Formulations 40
  4.4 Reducing The Size Of The Network 44
  4.5 The Single-Origin/Destination Network 45
  4.6 Multi-Origin/Destination Network 55
  4.7 Real Regular Matrix M 59
    4.7.1 Change In The Average Cost Of Travel 59
    4.7.2 Flow Redistribution 64
  4.8 Irregular Matrix M 69
    4.8.1 Flow Redistribution 70
    4.8.2 Change In The Average Cost Of Travel 73
  4.9 Conclusion 78
Chapter 5 (Varying-Demand Networks)

5.1 Introduction 81
5.2 Notation And Assumptions 82
5.3 Effect Of A Change In Demand On The Link Flow 88
5.4 Catastrophic Behaviour Of The Equilibrium Point 90
5.5 Conclusion 97

Appendix 98

Shortest Route Algorithms 99
Dijkstra 1 100
Dijkstra 2 103
Floyd 1 107
Floyd 2 108
Example 111
Alpath 113
Example 117
Useopt 119
Example 125
Detree 128
Example 131
Nnmrtpl 132
Example 138

References 140
Traffic Assignment and Network Analysis
by
N. Z. Simhairi

ABSTRACT

This thesis studies the transportation network, and is divided into three sections.

Initially an algorithm is described which finds the user-equilibrium assignment for networks with linear congestion functions where the cost of travel on a link is dependent on the flow in the whole network.

Secondly it investigates the sensitivity of the cost of travel and of the flow distribution in the network, to changes in the link congestion function. Combinatorial methods are used for evaluating the results of the sensitivity analysis. This is done with the aim of obtaining fast and efficient algorithms for the evaluation of cost sensitive and paradoxical links.

Finally, for networks where the demand is elastic, it describes the catastrophic behaviour of the point representing the user-equilibrium flow distribution under certain cost conditions.
CHAPTER 1
Introduction

Transportation is the transfer of goods or persons from one geographical location to another using the available modes of transport.

To study the transportation network and find ways of improving it, traffic engineers represent the transportation network by a graph consisting of a finite number of links and nodes. The nodes represent road intersections, residential areas, places of work and so on. The node from which traffic originates is called the origin and the node at which traffic terminates is called the destination. The arcs represent a one direction flow of traffic such as major roads, minor roads, railway lines and so on. The details and size of the graph vary according to the size and objectives of the study being conducted.

Two parts of the transportation network can be distinguished:

1- The demand for transport, formed by the persons and goods willing and able to travel from one node to another using the existing links. The demand for travel between any origin-destination pair is assumed to be either fixed or elastic. The latter is represented by a function reflecting the number of persons willing and able to travel at a given cost of travel or at a particular time.

2- The supply of travel, formed by the transport infrastructure and the cost of travel on the links (the link congestion function). The cost of travel on a link as perceived by the users will reflect the distance, time, comfort and price among other factors. Although perception of cost varies from person to person, the cost function reflects a general perception.
Various types of cost functions have been used. The main difference is whether the cost of travel on a link depends on the flow on that link alone or on the flow on other links too.

The next step for traffic engineers is to predict the traffic assignment; that is, the distribution of traffic on the links of the network. Two lines of thought are considered:

1- Traffic pattern is regulated by an external controller aiming to minimize the total travel cost in the network. In this case the resulting flow (assignment) is called the system-optimal flow (assignment).

2- Travellers choose their routes so that their travel cost is minimized. In this case the resulting flow (assignment) is called user-equilibrium flow (assignment).

Braess (1968) produced an example of a single origin-destination network showing that in the user-equilibrium assignment, some links behave in a paradoxical manner, that is, the removal (addition) of a link from (to) the network might cause a (an) decrease (increase) in the average cost of travel.

In chapter two, we give a more detailed introduction to the transportation network and the formulation of the traffic assignment according to the user-equilibrium and system-optimal assumptions. A brief run through some of the work conducted by various authors on the assignment problem is given.

This is followed, in chapter three, by the introduction of a simple algorithm for solving the user-equilibrium assignment for networks where the cost of travel on a link is linear, and is dependent on the flow on other links in
the network as well as the flow on the link itself. The example of Braess' paradox is given at the end of chapter three.

In recent years space and resources have become more and more limited and traffic engineers aim to obtain the maximum utility (reduction in total travel cost) from the allocation of given resources to network improvements. Network improvement can be effected in several ways, such as: adding new link to the network, reversing the direction of flow, applying slight modifications or even (paradoxically) removing existing links.

Since the link congestion functions are only known roughly, an algorithm for predicting the effect of a small change in the congestion function of a link on the total travel cost will assist traffic engineers in their decision making.

In chapter four, we carry out a linear sensitivity analysis on the assignment problem. By analysing the ratio of two determinants, we predict the effect of a change in the cost of travel on a link on the total travel cost and on the flow redistribution on other links in the network. We also identify the links that should give the largest decrease in travel cost if slight improvements are carried out. We give a combinatorial method to evaluate the determinants for the case where the cost of travel on a link is dependent on the flow on that link alone and the demand for travel is fixed. We then show that the coefficient of the change in the cost of travel on any link a on the flow on some link b is equal to the coefficient of the change in the cost of travel on link b on the flow on link a.

In chapter five the transportation network with an elastic demand for travel is considered. We show that the coefficient of the change in demand for origin-destination
pair \( w_i \) on the average cost of travel for origin-destination pair \( w_j \) is equal to the coefficient of the change in demand for origin-destination pair \( w_j \), on the average cost of travel for origin-destination pair \( w_i \). Then we show that the flow through the paradoxical links decreases as the demand for travel in the network increases. Finally, we show that under certain cost and demand conditions, the point representing the user-equilibrium flow will follow a standard fold catastrophe path.

Computer programmes in FORTRAN were written to find the user-equilibrium assignment as in chapter three, and for evaluating the effect of a change in the cost function on one link on the average cost of travel and on the flow on other links (as in chapter four). Programme listings are available in the appendix.

The graph \( N(V,L) \) is usually drawn from the geographical map of the area under study. Although it must reflect accurately the routes used, the exact topology of the map need not be followed, and it will appear later that this flexibility can be useful. For a large scale network, nodes portray important road intersections, large and small towns and villages. In some cases a number of small towns are combined to form one node, likewise a large town may be split into several nodes, each representing a certain section of the town. The links portray part of the existing road (and possibly rail) network, taking into account the type and width of each road and other relevant features. On the other hand, for a small scale model, the nodes will represent small residential districts or estates, small factories, local shopping areas, schools and so on. The links will portray minor local roads as well as major roads. The inclusion and exclusion of links from the graph is left to the discretion of the traffic engineer in charge. Such decisions are made in accordance with the aims and objectives of the conducted study.
CHAPTER 2

Transportation Networks

2.1 General Description

The transportation network can be represented by a graph $N(V,L)$ consisting of a finite number of nodes $V$, representing:
- Road crossings when it is possible to move from one road to the other.
- Points where the width or type of road is changing.
- Points of access from residential areas and places of work, and so on, and a finite number of links $L$, representing: a one-direction flow of traffic, from one node to another. This flow might be on a motorway, major or minor road, railway line, and so on.

The graph $N(V,L)$ is usually drawn from the geographical map of the area under study. Although it must reflect accurately the routes used, the exact topology of the map need not be followed, and it will appear later that this flexibility can be useful. For a large scale network, nodes portray important road intersections, large and small towns and villages. In some cases a number of small towns are combined to form one node; likewise a large town may be split into several nodes, each representing a certain section of the town. The links portray part of the existing road (and possibly rail) network, taking into account the type and width of each road and other relevant features. On the other hand, for a small scale model, the nodes will represent small residential districts or estates, small factories, local shopping areas, schools and so on. The links will portray minor local roads as well as major roads. The inclusion and exclusion of links from the graph is left to the discretion of the traffic engineer in charge. Such decisions are made in accordance with the aims and objectives of the conducted study.
Transportation is the transfer of goods and/or persons from one geographical location to another, that is, from one node to another, using the available transport means (modes), through the existing network of links. Although there are other approaches to the study of transportation — e.g., urban velocity fields, the study of effects of changes on a particular link requires the use of network analysis.

Two parts of the transportation problem can be distinguished: the supply of transport, and the demand for transport.

2.1.1 Demand

The demand for transport is formed by the persons and goods willing to travel from one node to another. The demand for travel could be static; fixed demand for travel between the various nodes, or an elastic demand; as a function of the cost of travel between the nodes. The demand over time is usually assumed to be constant, in some cases this is thought to be insufficient and a varying demand over time is used.

Therefore it is essential to identify the nodes where the persons and goods originate from (sources, origins), and the nodes where the persons and goods are destined for (sinks, destinations).

In the morning rush hour residential areas usually provide the nodes termed 'origins' and places of work or school provide the 'destinations'. The role of the nodes is reversed for the evening rush hour, although then the situation is more complicated.

Once the origin destination pairs are identified, then for any origin destination pair \((i,j)\), either a fixed demand \(d_{ij}\) is assumed between the two nodes (no change in demand with cost), or an elastic demand is assumed.
represented by a function reflecting the number of persons willing to travel between origin-destination pair \((i,j)\) for a variable cost of travel.

Several modes of transport could be in use in transferring persons and goods between the origin-destination pairs, such as cars, vans, lorries, buses, trains and so on. All modes in use must be noted and taken into account when calculating the cost of travel. The problem of varying demand over time will not be considered in this work.

### 2.1.2 Supply

The transport infrastructure forms the supply of transport, together with the cost of travel on the network. The cost of travel on a link as perceived by the users will depend on time, distance, comfort among other factors. It is either fixed, for uncongested networks; or, more probably, a function increasing with flow on the link itself or also with the flow on adjacent links. The cost of travel on a link in a congested network (termed the link congestion function), must have the following properties:

1. The cost should be positive.
2. It should increase with flow.
3. As the road capacity is reached, the cost must increase rapidly.

Here are some popular link congestion functions [from the National Co-operative Highway Research Program (1968)]

\[
  c_a(f_a) = \alpha_a \exp\left(\frac{f_a}{cap_a} - 1\right)
\]

\[
  c_a(f_a) = \alpha_a \left[ 0.87 + 0.13\left(\frac{f_a}{cap_a}\right)^4 \right]
\]

\[
  c_a(f_a) = \alpha_a \left[ 1 + 0.15\left(\frac{f}{cap_a}\right)^4 \right]
\]
To account for traffic congestion at road intersections, several authors [such as, Dafermos (1972, 1980), Smith (1979, 1983), Florian and Spiess (1981)], studied networks with the following type of cost function:

$$c_a(f) = \sum_{i \in \text{set of links}} \beta_i f_i + k_a$$

where $c_a(f_a)$ is the cost of travel on link $a$, per user, when the flow on link $a$ is $f_a$.

$c_a(f)$ is the cost of travel on link $a$, per user, when the flow distribution is $f$.

$a, \beta, k$ are constants.

$cap_a$ is the capacity for link $a$.

For rail transport the congestion function is constant for small flows and increases slowly until the flow reaches capacity level. The type of congestion function used depends entirely on the network studied.

2.2 The Traffic Assignment Model.

In recent years space has become more and more limited, and it is often argued that the construction or improvement of the transport infrastructure is one of the most important instruments of implementation of economic development policy. To tackle such a problem we must first find a way of predicting the traffic pattern on the network.

The prediction of traffic pattern, created by the distribution of demand among the various possible routes, joining the origin-destination pairs, is known as traffic assignment. Two lines of thought may be considered on the traffic assignment:
1. Traffic pattern is regulated by an external controller. Here the aim of the assignment is to minimize the total traffic cost in the network. The resulting flow pattern is called the system-optimal flow.

2. Passengers choose their routes so that their individual cost is minimized. The aim of the assignment is to reflect such passenger behaviours. The resulting flow pattern is called the user-equilibrium flow.

Notation:

Consider a transportation network \( N(V,L) \) consisting of \( V \) nodes and \( L \) links, where:

- \( c_{a}(f) \) is the cost of travel on link \( a \) per user when the flow distribution is \( f \).
- \( f_{a} \) is the total flow on link \( a \).
- \( f_{sa} \) is the flow on link \( a \) originating at node \( s \).
- \( W_{i} \) is the set of links originating at node \( i \).
- \( V_{i} \) is the set of links terminating at node \( i \).
- \( d_{ij} \) is the demand for travel between origin \( i \) and destination \( j \).
- \( Z \) is the number of origin-destination pairs.
- \( P_{W_{i}} \) is the set of routes joining O-D pair \( W_{i} \).
- \( \delta_{aj} \) = \( \{1 \) if link \( a \) belongs to route \( j \).
- \( \sum_{a} \delta_{aj} = 1 \) otherwise
- \( F_{j} \) is the flow on route \( j \).
is the average travel cost between origin-destination pair \( w_i \).

The formulation in what follows is based on a network with a fixed demand function and without explicit capacity constraints on the links.

2.2.1 System-Optimal Flow

The assignment of flows to the network is such that the total cost of travel in the network is minimized. The formulation of the problem is

\[
\min \sum_a c_a(f_a) \cdot f_a \quad \text{(2.1.)}
\]

subject to

\[
\sum_s f_a^s = f_a \quad a = 1, \ldots, L \quad \text{(2.2)}
\]

\[
f_a^s \geq 0 \quad a = 1, \ldots, L; \quad s = 1, \ldots, q \quad \text{(2.3)}
\]

\[
\sum_{a \in W_i} f_a^s - \sum_{a \in V_i} f_a^s = h_i^s \quad i = 1, \ldots, V; \quad s = 1, \ldots, q \quad \text{(2.4)}
\]

where

\[
h_i^s = \begin{cases} 
- \frac{d_{is}}{d_{sj}} & \text{if } i \text{ is a destination node} \\
\sum_j d_{sj} & \text{if } i = s \\
0 & \text{otherwise}
\end{cases}
\]

This type of transportation assignment can be applied for problems where the traffic pattern is regulated by a central authority; for example, rail transport; the transportation of military supplies; or the supplies for a large manufacturing industry.

2.2.2 User-Equilibrium Flow

In the normal road traffic network, individual users choose their routes with the aim of minimizing their own cost. This was observed and discussed by Wardrop (1952),
and stated as his first principle: "The journey times on all the routes actually used are equal, and less than those which would be experienced by a single vehicle on any unused route". Therefore an individual would only change his route to a cheaper route.

The assignment is to find a flow pattern satisfying the following:

\[
\text{If } \sum_{a=1}^{L} c_a(f) \delta a_j = U_{w_i} \text{ then } F_j \geq 0, \quad i=1...Z \quad \text{(2.5)}
\]
\[
\& \text{ if } \sum_{a=1}^{L} c_a(f) \delta a_j < U_{w_i} \text{ then } F_j = 0, \quad i=1...Z \quad \text{(2.6)}
\]

Also constraints (2.2), (2.3) and (2.4) must hold.

Beckman et al (1959), showed that, in the case where \( \frac{\partial f_a}{\partial f_b} \) is symmetric, the user-equilibrium flow assignment can be formulated as a convex optimization problem subject to flow conservation constraints. The user optimal flow is the variational problem:

\[
\min \quad \int \sum_{a} c_a(f) \, df \quad \text{ ................. (2.7)}
\]

subject to

\[
\sum_{s=1}^{q} f_a^s = f_a \quad a = 1 ... L \quad \text{ ............. (2.8)}
\]

\[
f_a^s \geq 0 \quad a = 1...L; \quad s = 1...q \quad \text{ ....... (2.9)}
\]

\[
\sum_{a \in W_i} f_a^s - \sum_{a \in W_i} h_i^s, i = 1..V; \quad s = 1...q \quad \text{ ... (2.10)}
\]

2.3 Traffic Assignment Techniques

If the network is uncongested, so that the link cost (congestion) function is constant, then the user-equilibrium and system-optimal flow distributions will be the same. Network users will simply use the shortest route between each origin-destination pair of nodes.
Steenbrink (1974) gives a description of several shortest route algorithms. Two such algorithms are: Dijkstra's algorithm and Floyd's algorithm. These two algorithms are programmed in Fortran to find the shortest routes in a multi-mode user network, the London Underground system is used as a model of such a network [for programme listing and comments, see the appendix].

In the road traffic network the congestion function is usually an increasing function with flow. The formulation of the assignment and production of algorithms to solve it by various authors throughout the years have differed in respect of two main factors:

a) The congestion function: Initially the congestion function used was of the form \( c_a(f_a) \); where the cost of travel on link \( a \) is dependent on the flow on that link alone. In order to account for two way traffic and the congestion caused at road intersections, an extended form of the congestion function was introduced, \( c_a(f) \); where the cost of travel on link \( a \) depends on the flow in the whole network. In practice \( c_a(f) \) depends on the flow in a few links other than link \( a \). The latter congestion function coupled in some cases with a multi-mode user-networks led to networks where the Jacobian matrix \( \frac{\partial c_a}{\partial f_b} \) is not symmetric [Asymmetric assignment].

b) The demand function: Most work was conducted on the assumption that the demand for flow between the various origin-destination pairs was constant over time and average cost of travel. Other work assumed elastic demand; as the cost of travel decreases (increases) the demand increases (decreases). Finally a smaller number of authors tackled the problem of varying demand over time.
After the initial formulation of the user-equilibrium problem as a constraint minimization problem by Beckman et al (1959), many authors tackled the problem of finding an efficient algorithm to solve it. We start with some authors who assumed that the demand function is constant. Dafermos and Sparrow (1969) formulated the user-equilibrium and system-optimal assignment, and gave an algorithm to solve the assignment for a network where the congestion function takes the form, \( c_a(f_a) \). Dafermos (1971) extended the above model to include congestion functions of the form \( c_a(f) \), where the existence, uniqueness and stability of the solution is subject to the Jacobian matrix \( \frac{\partial c_a(f)}{\partial f} \) being symmetric (for the user-equilibrium flow assignment). Dafermos (1972) extended the model further to accommodate networks with multi-mode users. Leventhal et al (1973) used a column generation algorithm to find the user-equilibrium and system-optimal solution, and compared this algorithm with that introduced in Dafermos and Sparrow (1969). Nguyen (1974) decomposed the problem into a set of simpler subproblems, then used an adaptation of the convex simplex method to solve each sub-problem. Leblanc et al (1975), also decomposed the problem by linearising the objective function and used a Frank-Wolfe algorithm to calculate the feasible direction. In using this technique none of the constraints need be considered explicitly; they are satisfied automatically. Both above authors use \( c_a(f_a) \) as a link congestion function. Daganzo (1977, a,b), showed that Nguyen's (1974) and Leblanc's et al (1975) algorithms could be extended to apply to the network with explicit capacity constraints, but needed a strong condition to do so. Hearn et al (1981) showed that Daganzo's modified Frank-Wolfe algorithm converges under a weaker condition.

In networks where the effect of the flow on link a, on the cost of travel on any link b differs from the effect of flow on link b, on the cost of travel on link a, asymmetric jacobian matrices may result. Smith (1979 b)
proved the existence, uniqueness and stability of the solution to the asymmetric network under certain conditions. Dafermos (1980) showed that the conditions imposed by Smith (1979b) are equivalent to: The jacobian matrix is everywhere positive definite, and gave an algorithm to solve the problem. Smith (1983a,b) gave algorithms to find the equilibrium flow for the network described by him in 1979. Florian and Spiess (1981) gave a diagonalization algorithm to solve the asymmetric assignment problem; the convergence of the algorithm is then assured under certain conditions.

Beckman et al (1959), in their formulation, allowed for the demand function to be a decreasing function of the cost of travel. Wilkie and Stefanek (1971) used two techniques: the Newton-Rapson and constrained gradients, to solve the assignment problem with congestion functions of the form $c_a(f_a)$ and varying demand. Fisk and Nguyen (1979) discussed the existence and uniqueness of a solution to the assignment problem with flow dependent transit costs, and the convergence of some algorithms. Dafermos (1982) discussed two algorithms for solving asymmetric assignment networks with elastic demand with cost, and gave the condition for convergence of both algorithms. Fisk and Nguyen (1982) discussed several proposed algorithms for solving the asymmetric cost networks, and compared their performance in solving a small realistic network, with a fixed demand.

Merchant and Nemhauser (1978) looked into a network with a single destination, and a congestion function of the form $c_a(f)$ and a varying demand over time. The problem is presented as a nonconvex mathematical programming problem. Fisk (1980) also discussed a network with elastic demand over time, and a congestion function of the form $c_a(f_a)$, then showed the similarities between fixed demand incremental assignment techniques and an algorithm for solving the stochastic model.
In the following chapter we introduce an algorithm for finding the user-equilibrium flow in any network, where the cost of travel on a link is linear and depends on the flow distribution in the whole network.

In the previous chapter, the formulation of the traffic assignment problem was given according to the linear cost and user equilibrium assumptions of Wardrop (1952). Some algorithms for solving both problems were described in. In this chapter we introduce a simple algorithm for solving the traffic assignment problem assuming in the user-equilibrium assumption. The possible equilibrium behaviour of the links in the network is described.

For any transportation network \( W(V,E) \), let \( p \) be the number with the shortest distance between origin-destination pair \( e \). The distance travelled on the set of used routes between origin-destination pair \( e \) should differ only by a small percentage from the distance travelled on route \( p \). For example, Wright (1976) shows that, in central London, the average distance travelled is only 5% in excess of the shortest distance route. Therefore, when using traffic assignment algorithms based on advance knowledge of the set of routes joining all origin-destination pairs, such as those of Dafermos (1971, 1972), it is not necessary to find all such routes; obtaining the routes that are within a certain percentage of the shortest distance is sufficient.

5.2 Finding Plausible Used Routes

The following algorithm finds a sufficient set of routes for determining the traffic assignment. The algorithm is based on Dijkstra's shortest route algorithm and a branch and bound technique on the successive removal of links from the shortest route; the algorithm operates as follows:
CHAPTER 3

Traffic Assignment

3.1 Introduction

In the previous chapter, the formulation of the traffic assignment problem was given according to the system-optimal and user-equilibrium assumptions of Wardrop (1952). Some algorithms for solving both problems were referred to. In this chapter we introduce a simple algorithm for solving the traffic assignment problem according to the user-equilibrium assumption. The possible paradoxical behaviour of the links in the network is mentioned.

For any transportation network $N(V,L)$, let $p$ be the route with the shortest distance between origin-destination pair $w^1$. The distance travelled on the set of used routes between origin-destination pair $w^1$ should differ only by a small percentage from the distance travelled on route $p$. For example, Wright (1976) shows that, in Central London, the average distance travelled is only 5\% in excess of the shortest distance route. Therefore, when using traffic assignment algorithms based on advance knowledge of the set of routes joining all origin-destination pairs, such as those of Dafermos (1971, 1972), it is not necessary to find all such routes; obtaining the routes that are within a certain percentage of the shortest distance is sufficient.

3.2 Finding Plausible Used Routes

The following algorithm finds a sufficient set of routes for determining the traffic assignment. The algorithm is based on Dijkstra's shortest route algorithm and a branch and bound technique on the successive removal of links from the shortest route; the algorithm operates as follows:
For O-D pair \( w_i \) of network \( N \)

Let \( P^* = (a_1, a_2, \ldots, a_n) \) be the shortest distance route for O-D pair \( w_i \).

Form a new network \( N(b, j, x_j) \) such that

\[
N(b, j, x_j) = N - \text{link } p_{b,j}
\]

\[
b + 1
\]

\[
b > r \quad \text{no}
\]

\[
b = 1
\]

Find the shortest distance route for O-D pair \( w_i \) of network \( N(b, j, x_j) \), call it \( P(b, j, x_j) = (a_1, a_2, \ldots, a_{i_j}) \), where

\[
p(b, j, 1) = a_1, p(b, j, 2) = a_2, \ldots, p(b, j, i_j) = a_{i_j}
\]

where \( i_j \) is the number of links in the route.

\[
\text{does } P(b, j, x_j) \text{ exist and}
\]

\[
\text{the cost of travel on } P(b, j, x_j) < \kappa (\text{the cost of travel on } P_j)
\]

where \( \kappa \) is the increase in the cost of travel allowed.

\[
\text{yes}
\]

\[
x_j + 1
\]

\[
x_j > i_{(j-1)} \quad \text{no}
\]

\[
j + 1
\]

\[
x_j + 1
\]

Form a new network \( N(b, j, x_j) \), such that

\[
N(b, j, x_j) = N(b, j-1, x_{(j-1)}) - \text{link } p(b, j-1, x_j)
\]

\[
j - 1
\]

\[
j = 1 \quad \text{no}
\]

\[
x_j + 1
\]

\[
\text{yes}
\]

\[
b + 1
\]

\[
b > r \quad \text{no}
\]

\[
\text{STOP}
\]
At each stage, Dijkstra's algorithm is used to find the shortest route between the origin-destination pair. This algorithm is programmed in FORTRAN on the VAX 11/780. Programme listing and an example are available in the appendix.

For networks with several origin-destination pairs the above process is repeated for each origin-destination pair successively.

The assumption in Wardrop's first principle of user-equilibrium assignment is that: all users have total knowledge of all the routes (within a reasonable distance from the shortest route), and the cost of travel on such routes, joining their origin node to their destination node. The assumption of total knowledge of all such routes might not always be true. It is quite conceivable that the travellers between some origin-destination nodes do not have the knowledge of certain routes between their origin-destination pair which are cheaper to use. This is especially likely for long-distance travellers where cheaper routes can be achieved by the use of local traffic networks somewhere between their origin-destination pair. For example, consider the network N(4,6), with two origin-destination pairs \((0_1,D_1)\) and \((0_2,D_2)\), shown in Fig 3.1.

Assume that travellers between \((0_1,D_1)\) know only the following two routes:

Route 1 = (link a)
Route 2 = (links b, c, d).

Figure 3.1
And travellers between \((0_2, D_2)\) know routes:

Route 3 = (link f), route 4 = (link e) and route 5 = (link c).

Now if the cost of travel along route 1 is equal to that along route 2 and if the cost of travel along route 3 is equal to that along route 4 but less than that along route 5, a steady user-equilibrium will be achieved. However, if the complete knowledge of routes is assumed, then travellers between \((0_1, D_1)\) will know of route 6 = (links b, e, d) and route 7 = (links b, f, d), which are cheaper to travel along than route 2. This will inspire travellers to change from using route 2 to using routes 6 and 7, resulting in a new flow distribution. Then either the cost of travel on routes 3, 4 and 5 and on routes 1, 2, 6 and 7 will be equal, or route 2 will be replaced by routes 6 and 7, leaving link c unused.

If such a situation occurs in practice, then algorithms (for finding the traffic assignment) that require the prior knowledge of the relevant routes have an advantage over other algorithms. By excluding certain routes from the beginning of the assignment process, traffic engineers can achieve a traffic assignment closer to what they believe occurs in reality.

3.3 An Algorithm For Finding the User-Equilibrium Assignment.

Suppose we are given a multi-origin/destination network. For any origin-destination pair \(w_i\) with \(q\) routes joining them, the user-equilibrium assignment is achieved when the routes have been numbered such that:

\[
C_1 = C_2 = C_3 \ldots \ldots = C_n \leq C_{n+1} \ldots \ldots C_q
\]

when

\[
F_1, F_2, \ldots, F_n > 0 \text{ and } F_{n+1}, \ldots, F_q = 0,
\]
where $C_i$ is the cost of travel on route $i$, and $F_i$ is the flow on route $i$.

The following algorithm is designed to obtain the user-equilibrium assignment for a multi-origin/destination network, where the demand for travel is fixed. It finds the set $P_{w_i}$ of routes actually used between origin-destination pair $w_i$ and the flow on each route.

1. For each pair $w_i$, introduce the shortest distance route that joins the two nodes to $P_{w_i}$.

2. Find the number of travellers on each route in $P = \{P_{w_i}\}$, satisfying the conditions that:
   
   i) $C_i = C_j$ for all $i$ and $j$ belonging to $P_{w_i}$,
   
   ii) the sum of flows on all routes $j$ belonging to $P_{w_i}$ is equal to the demand for travel between origin-destination pair $w_i$.

3. If the flow on any route $j$ belong to $P_{w_i}$ is then $< 0$, remove route $j$ from $P_{w_i}$ and go to step 2.

4. Find the cost of travel for all origin-destination pairs on the used routes $U_{w_i}$.

5. For each origin-destination pair $w_i$ find the route $s \notin P_{w_i}$ with $C_s = \min C_k$ over routes $k$ joining origin-destination pair $w_i$ but not in $P_{w_i}$.

6. If $C_s < U_{w_i}$, then introduce route $s$ to $P_{w_i}$.

7. If a new route has been introduced to $P$ go to step 2. Otherwise stop here.
Convergence of The Algorithm.

If the cost functions are symmetric - that is, if
\[
\frac{\partial c_a(f)}{\partial f} = \frac{\partial c_b(f)}{\partial f},
\]
where \( \frac{\partial c_a(f)}{\partial f} = \frac{\partial c_b(f)}{\partial f} \), it is possible to find a convex function \( S^*(F) \) such that
\[
\frac{\partial S^*(F)}{\partial F_i} = C_i(F)
\]
(Dafermos 1971). Therefore the user-equilibrium traffic assignment can be reformulated as the following convex optimization problem,

\[
\min S^*(F)
\]
such that
\[
\sum_{j \in P_j} F_j = d_{w_i}, \quad \text{and} \quad F_j \geq 0.
\]

Assume that at iteration \( L \), in the above algorithm, the set of routes is \( P = \{P_i, P_j\} \). If route \( i \in P_i \) then \( F_i > 0 \) and if route \( j \in P_j \), then \( F_j = 0 \). We first prove a useful lemma.

**Lemma**

The constrained gradient vector of \( S^*(F) \) with respect to a small change in \( F_j \), termed \( \nabla_j S^* \), is given by

\[
\nabla_j S^* = \left( \frac{\partial S^*}{\partial F_j} - U_{w_i} \right)
\]

where \( w_i \) is the origin-destination pair joined by route \( j \).

Taking a linear approximation, a change in the flow of any route \( j \) in \( P_j \) from zero flow to some small \( \delta F_j > 0 \)
causes the objective function to be

$$S^*(F^*) = S^*(F) + \frac{\partial S^*(F)}{\partial F_j} \delta F_j + \sum_{i \in P_j} \frac{\partial S^*(F)}{\partial F_i} \delta F_i$$  \hspace{1cm} (3.1)$$

where $F^*$ is the new flow distribution resulting from the introduction of route $j$ to $P_j$.

Now $$\nu_j S^* = \lim_{\delta F_j \to 0} \frac{S^*(F^*) - S^*(F)}{\delta F_j}$$  \hspace{1cm} (3.2)$$

and we have $$\sum_{k \in P_{w_j}} \delta F_k = 0$$  \hspace{1cm} (3.3)$$

for all $w_j \neq w_i$ where $w_i$ is the origin-destination pair joined by route $j$, and

$$\sum_{k \in P_{w_i}} \delta F_k = - \delta F_j$$  \hspace{1cm} (3.4)$$

From above, we also know that $$\frac{\partial S^*}{\partial F_i} = U$$ for the corresponding origin-destination pair.

Therefore, from equations (3.1), (3.2), (3.3) and (3.4),

$$\nu_j S^*(F) = \frac{\partial S^*(F)}{\partial F_j} - U_{w_i}$$

that is $$\nu_j S^*(F) = C_j(F) - U_{w_i}$$

This completes the proof of the lemma.

If the cost of travel, $C_j$, on a route $j$ where $F_j = 0$ and $j$ joins $O-D$ pair $w_i$, is less than $U_{w_i}$, then $\nu_j S^* < 0$. In this case introducing route $j$ to the set of used routes will cause $F_j$ to increase and hence $S^*(F)$
will decrease. The increase in flow on route $j$, joining origin destination pair $w_i$, is restricted by two factors:

a) the constrained gradient $v_j S^*$ must remain $\leq v_i S^*$ for any route $i \in P_i$;

b) the flows on all routes must remain $\geq 0$.

Hence, introducing the route with a cost lower than the average cost of travel from its origin to its destination causes the objective function to decrease at each step. Since $S^*(F)$ is convex, and there are a finite number of routes joining the origins to the destinations of the network and the value of $S^*(F)$ decreases at each step, the algorithm will converge to the user-equilibrium solution.

We consider the change in $S^*$ in one iteration on the larger loop marked on flow chart 1, resulting from the addition of one route to $P_{w_i}$. After each iteration on this loop the smaller loop is used to make the new solution feasible, and we compare $S^*(F)$ at two successive visits to step 6.

A computer programme has been developed using the algorithm to calculate the user-equilibrium flow distribution for a network where the cost of travel on a link is linear and dependent on the flow distribution in the whole network. A standard algorithm is used for solving the linear matrix equation $Bx = b$, where :-
1. For origin-destination pair \( w_i \), find the shortest route joining the origin to the destination, add it to \( P_{w_i} \).

2. Find the flow distribution given that the set of used routes is \( P_{w_i} \).

3. Any route \( j \in P_{w_i} \) with \( \frac{c_{wj}}{f_j} < 0 \)?
   - Yes: remove route \( j \) from \( P_{w_i} \).
   - No: add route with least cost to \( P_{w_i} \).

4. Find the cost on used routes.

5. Find cost of unused routes.

6. Is cheapest unused route < cost of travel on used routes?
   - Yes: STOP
   - No: flow chart 1.
B is an \((m + z) \times (m + z)\) matrix

\(m\) is the number of used routes

\(z\) is the number of origin-destination pairs

\(b\) is an \(m \times z\) element vector

The cost of travel on any route \(i\) in \(P\) is:

\[ C_i = \sum_{j} a_{ij} F_j + k_i. \]

Then, the set of equations forming \(Bx = b\) at iteration \(i\) is

\[ \sum_{j \in P_{w_i}} a_{ij} F_j - U_{w_i} = -k_i \text{ for } i = 1, 2, \ldots, m_z \text{ and all O-D pairs } w_i \]

and

\[ \sum_{j \in P_{w_i}} F_j = d_{w_i} \text{ for all O-D pairs } w_i \]

where \(m_z\) is the number of used routes at iteration \(i\).

A solution \(F_i\) (flow distribution at iteration \(i\)) is always obtainable providing that the matrix \(B_i\) (matrix \(B\) at iteration \(i\)) is not singular. In the single O-D case, if matrix \(B_i\) (with the exception of its last row and last column) is diagonally dominant, then matrix \(B_i\) is unlikely to be singular. The assumption of diagonal dominance is reasonable, since the cost of travel on any route is likely to be more strongly dependent on the flow along the route itself than on the flow along other routes.
If the cost function is asymmetric - that is
\[ \frac{\partial C_a(f)}{\partial f_b} \neq \frac{\partial C_a(f)}{\partial f_b} \], it is not possible to find a convex
function \( S^*(F) \) such that \[ \frac{\partial S^*(F)}{\partial F_i} = C_i(F) \], nor is it
possible to formulate the user-equilibrium assignment as
a convex optimization problem. In the asymmetric case,
the cost function can be represented in the following
matrix form:

\[ C(F) = GF + DF \]

where \( G \) is symmetric.

If matrix \( G \) dominates matrix \( D \) - that is, the cost
function is not too asymmetric, the above algorithm will
most probably converge, due to the close resemblance to
the symmetric case. However, if the cost function is
asymmetric with matrix \( G \) not dominant over matrix \( D \),
then cycling might occur, hence preventing the algorithm
from converging. The following example highlights this
case.

Example

Let network \( G(V,L) \) (figure 3.3), be a single origin
destination network containing four routes with the following
cost functions.

\[ C_1(F) = 2F_1 + 2F_2 + 4F_3 \]
\[ C_2(F) = F_1 + 6F_2 + 2F_3 \]
\[ C_3(F) = F_1 + 2F_2 + 3F_3 + 5F_4 \]
\[ C_4(F) = 4F_1 + 2F_2 + 2F_3 + 3F_4. \]
Let the demand from O to D be 8 units. Choosing route 2 as an initial solution, the flow distribution and average cost of travel will be:

a) \( F = (0, 8, 0, 0) \), \( U = 48 \) hence \( C(F) = 16 < U \).

Introducing route 1 to \( F \) gives:

b) \( F = (\frac{32}{5}, 0, 0, 0) \), \( U = 16 \)

here \( C(F) = \frac{48}{5} < U \), hence introducing route 3 to \( F \) gives:

c) \( F = (0, \frac{8}{5}, \frac{32}{5}, 0) \), \( U = 22.4 \)

here \( C(F) = 16 < U \), hence introducing route 4 to \( F \) gives:

d) \( (0, \frac{24}{7}, 0, \frac{32}{7}) \), \( U = 20.57 \)

here \( C(F) = \frac{48}{7} < U \), hence

e) \( F = (\frac{32}{5}, \frac{8}{5}, 0, 0) \), \( U = 16 \) as in step b.

Therefore the algorithm will cycle by repeating steps b, c, d continuously.

The above algorithm has been programmed in FORTRAN on the VAX 11/780 and a programme listing is given in the appendix.
3.4 Paradoxical Behaviour Of The Links

Due to the nature of the user-equilibrium assignment solution, the removal (addition) of a link from (to) the network may result in the decrease (increase) of the average cost of travel in the network. This was first pointed out through an example by Braess (1968) and later elaborated by Murchland (1970).

Braess' Example

Figure 3.2 shows a network with one origin-destination pair, four links and four nodes. Let the demand for travel between \((o,d)\) be 6 units, and the cost of travel on each link be as follows:

- Link \((o,a)\); \(c_1 = 10f_1\).
- Link \((a,d)\); \(c_2 = f_2 + 50\).
- Link \((o,b)\); \(c_3 = f_3 + 50\).
- Link \((b,d)\); \(c_4 = 10f_4\).

Two routes join \(o\) to \(d\):
- route \(p_1 = oad\)
- route \(p_2 = obd\)

The user-equilibrium solution has 3 units travelling on route \(p_1\) and 3 units travelling on route \(p_2\), the average cost of travel on each route is:

\[
C(p_1) = (10 \times 3) + 50 + 3 = 83
\]
\[
C(p_2) = 50 + 3 + (10 \times 3) = 83.
\]

Hence the average cost of travel between \(o,d\) is \(U_1 = 83\).
Now consider the same network with the addition of a new link \((a,b)\) with a cost function: \(c_5 = f_5 + 10\). This addition results in the creation of a third route \(p_3 = oabd\) (see Fig. 3.3).

The new user-equilibrium solution has 2 units travelling along each of the three routes, with the cost of travel on each route equal to:

\[
C(p_1) = (4 \times 10) + 2 + 50 = 92 \\
C(p_2) = 2 + 50 + (4 \times 10) = 92 \\
C(p_3) = (4 \times 10) + 2 + 10 + (4 \times 10) = 92
\]

Figure 3.3

hence the average cost of travel between \((o,d)\) is \(u_2 = 92 > u_1\). The addition of link \((a,b)\) results in an increase in the average cost of travel between nodes \((o,d)\) equal to \(9, (92 - 83)\).

The link \((a,b)\) is termed a "Paradoxical Link". Knodel (1969) reported that Braess' paradox may have occurred in practice in the city of Stuttgart; where the elimination of a link (representing the lower part of Konigstrasse) resulted in the reduction of congestion.

Stewart (1979) gave examples to show how the total cost of travel can be reduced by introducing restrictions to the number of units travelling on the links. Traffic engineers often use link elimination or capacity restrictions as a strategy to reduce congestion. In the following chapter we introduce a linear sensitivity analysis to detect the effect of small changes in the link congestion function on the flow and total cost of travel in the network.
CHAPTER 4

Linear Sensitivity Analysis to Detect Paradoxical and Cost Sensitive Links.

4.1 Introduction

Traffic planning engineers aim to obtain the maximum utility (reduction in total travel cost) from the allocation of given resources to network improvement. The link congestion function, relating the cost of travel on a link to the volume using the link, may only be known roughly, and the assignment is known to be sensitive to the choice of function, (Boyce, Janson and Eash (1981)).

Network improvement can be in the form of altering the congestion function on existing links, the addition of new links or reversing the direction of flow on existing links. Hence a procedure or algorithm for predicting the effect of improving an existing link, or adding a new link to the network, on the total cost of travel, will assist traffic engineers in their decision making. Frank (1980) discussed thoroughly the example of Braess (1968), and gave conditions for the existence of such paradoxical links. Smith (1978) showed how an increase in the cost of travel of a link, for a network with uncongested links and congested nodes, may decrease the average travel cost on the network. Frank (1984) studied the ladder network with a linear cost function, and showed that by imposing a simple marginal cost ratio on the links we can insure the non-existence of paradoxical links. Leblanc (1975) and Foulds (1981 and 1985) provided a branch and bound method for the reduction of traffic congestion and travel cost. Smith (1979a) and Gartner (1980 a) showed that by charging every user a toll equivalent to the marginal social cost, an optimal flow distribution can be reached.

Jarvis and Martinez (1977) gave a heuristic method for
the addition of a link to the network. Steinberg and Zangwill (1983) studied the effect of introducing a new path in a network where the travel cost of a link is dependent on the flow on that link alone, given a "user equilibrium" steady state traffic assignment. Dafermos and Nagurney (1984) extended this work to include networks with the travel cost on a link dependent on the flow on other adjacent links, and considered the effect of an increased demand on the total travel cost. Both pairs of authors represented their solution as a ratio of two large determinants.

In this section, a linear sensitivity analysis is carried out by analysing the ratio of two determinants similar to those derived by Steinberg and Zangwill (1983), in order to achieve the following objectives:

1. Identification of the links having a "Paradoxical effect" on the network - that is, where the introduction of a delay, or total removal, would lessen the total travel costs.
2. Identification of the links which give the largest decrease in travel cost if slight improvements are carried out.
3. Some measure of the sensitivity of a solution to the link congestion function used.

We give a combinatorial method for evaluating the determinants in the case where the link congestion function is dependent on the flow on that link alone. A computer program is given to evaluate the results. In the single origin-destination case the problem is reduced to that of finding spanning trees of the network with certain properties, and standard algorithms exist for this problem. The method is a modification of one well-known in electrical circuit theory (see Seshu and Reed (1961), Percival (1953), Berge (1966)). Indeed, for the single origin-destination network, the evaluation of the "denominator determinants" is effectively a result of Maxwell (1892).
4.2 Notation And Assumptions

We consider traffic flow on a directed graph \( N(V,L) \), where the set of links \( L \) contains \( k \) elements, with \( Z \) origin-destination pairs, each with a fixed and known demand, \( d_i \).

We shall assume that the link congestion function, \( c_a(f_a) \), is monotonically increasing and is dependent only on the number of vehicles on the link itself.

The following notation will be used:-

- \( N^*(V,L) \) is the undirected graph corresponding to \( N(V,L) \).
- \( a \in L \) is a link.
- \( f_a \) is the number of vehicles per unit time using link \( a \).
- \( f = \{ f_a \} \) is the distribution of vehicles on the links.
- \( c_a(f_a) \) or \( c_a \) is the link congestion function and is the cost of travel on link \( a \) for each vehicle, if the flow is \( f_a \); we shall assume that \( c_a \) is a monotonically increasing function of \( f_a \).
- \( c = \{ c_a \} \) is the link cost vector in \( \mathbb{R}^k \).
- \( g_a = \frac{c_a}{c_a'(f_a)} \); \( g_a > 0 \), from the last assumption.
- \( p, q \) are routes; sets of links forming a path from origin to destination.
- \( \delta_{ap} = \begin{cases} 1 & \text{if link } a \text{ is included in route } p. \\ 0 & \text{otherwise.} \end{cases} \)
\( F_p \) is the number of vehicles per unit time using route \( p \).

\( F = \{ F_p \} \) is the distribution of vehicles on the routes in the network.

\( W = \{ w_i \} \) is the set of all origin-destination pairs.

\( P = \{ P_{w_i} \} \) is the set of all independent routes joining O-D pair \( w_i \).

\( P \) is the set of all independent routes for which \( F_p > 0 \): assume that \( P \) has \( Q \) elements joining all O-D pairs – see below for the meaning of "independent" here.

\( C_p \) is the cost of travel on route \( p \), if the flow or \( C_p(F) \) distribution is \( F \).

\( C(F) = \begin{pmatrix} C_1(F) \\ \vdots \\ C_Q(F) \end{pmatrix} \) is the route cost vector in \( \mathbb{R}^Q \).

\( \mu_{w_i} \) is the cost of travel per vehicle, between O-D pair \( w_i \).

\( U_w = \begin{pmatrix} U_{w_1} \\ U_{w_2} \end{pmatrix} \) is a vector in \( \mathbb{R}^Z \).

\( U \) is the total cost of travel in the network.

\( d_{w_i} \) is the demand for travel for O-D pair \( w_i \).

\( \underline{d} = \{ d_{w_i} \} \) is set of all demands \( d_{w_i} \).

\( A \) is a \( k \times Q \) link-route incidence matrix, whose \((a,q)\) entry equals \( \delta_{aq} \).
B is a Z x Q matrix whose (i,p) entry equals 1 if route p joins 0-D pair \( w_i \) and 0 otherwise.

For a graph \( N \) with \( V \) links and \( n \) independent routes \( q \), joining origin-destination pair \( w_i \),

let \( P_q = \{ s_{1q}, s_{2q}, \ldots, s_{vq} \} \).

i.e. the qth column of matrix A.

The set \( P_{w_1} = \{ P_1, P_2, \ldots, P_q \} \) of all routes is dependent if there exists numbers \( \alpha_1, \alpha_2, \ldots, \alpha_q \) not all zero such that

\[
\sum_{i=1}^{q} \alpha_i \cdot P_i = 0.
\]

Otherwise the set \( P_{w_i} \) is independent.

Definition

A cycle is a finite sequence of links \( (a_1, a_2, \ldots, a_n) \) in which each link \( a_k \) has one node in common with the preceding link \( a_{k-1} \), and the other with the succeeding link \( a_{k+1} \); also the initial node \( (a_1) \) and terminal node \( (a_n) \) are the same.

The set of cycles \( \{ c_{y_1}, c_{y_2}, c_{y_3}, \ldots, c_{y_n} \} \) is dependent if there exists numbers \( \alpha_1, \alpha_2, \ldots, \alpha_n \) not all zero such that

\[
\sum_{i=1}^{n} \alpha_i \cdot c_{y_i} = 0,
\]

otherwise the set of cycles is independent.

As an example to see what "independent" means, consider a single-origin single-destination graph \( N(3,4) \) (see Fig 4.1).
Let $P = \{P_1, P_2, P_3, P_4\}$ be the set of used routes of $N$, with flow $F = \{F_1, F_2, F_3, F_4\}$. Then route 1 has a flow of $f_1$ and contains links 1,3; hence $P_1 = (1,0,1,0)$, similarly $P_2 = (0,1,0,1)$, $P_3 = (1,0,0,1)$ and $P_4 = (0,1,1,0)$. However by assigning $\alpha_1 = 1$, $\alpha_2 = 1$, $\alpha_3 = -1$ and $\alpha_4 = -1$, we find that:

$$\alpha_1 P_1 + \alpha_2 P_2 + \alpha_3 P_3 + \alpha_4 P_4 = 0.$$ 

Hence $P = \{P_1, P_2, P_3, P_4\}$ is dependent, and one of those routes must be removed.

Suppose that (say) $\min \{F_1, F_2, F_3, F_4\} = F_2$. Since $P_2 = P_3 + P_4 - P_1$, put $F^*_1 = F_1 - F_2$, $F^*_3 = F_3 + F_2$, $F^*_4 = F_4 + F_2$. This allocation to routes leaves the flow on each link unchanged, but only three independent routes are now used. The number of independent routes for a single O-D graph = the number of independent cycles + 1..(4.1)

For a graph $G$ with $Z$ origin-destination pairs, $K$ links and $V$ nodes the total number of independent routes (used and unused) = total number of independent cycles + $Z \leq K-V+1+Z$, where $K-V+1$ is the maximum number of independent cycles.

### 4.3 Formulations

Assume a user-equilibrium flow $F$ exists on the network. Then the cost of travel for origin-destination pair $w_i$ is

$$U_{w_i} = C_p(F) = \sum_a c_a(f_a) \text{ for all } p \in P_{w_i} \text{ and } F \geq 0.$$
In matrix form

\[ C(F) = A^T C(f) \]

[In addition \( \sum_{a} \delta_{pa} c_{a}(f_{a}) \geq U_{w_{i}} \) for all \( p_{i} \in P \) and \( F_{p} = 0 \), but this is not needed subsequently].

For each origin-destination pair, the sum of the number of vehicles on each route is the total demand \( \sum_{p_{i} \in P_{w_{i}}} F_{p} = d_{w_{i}} \), which can be written in matrix form as

\[ d = BF \]  .................................................. \( (4.2) \)

Now assume that the link congestion function for some link \( e \), \( c_{e}(f_{e}) \), is changed to \( c_{e}(f_{e}) + \varepsilon_{e} \), and that the used routes remain the same.

This will trigger off the following:

i) A change in the flow pattern of the network.

\[ F \to F + \delta F = F^{*} \]  (say)

Now (in matrix form)

\[ F = A^{T} \delta f \]  .................................................. \( (4.3) \)

so that if \( \delta f \to \delta f' = f^{*} \), then

\[ \delta F = A^{T} \delta f \]  .................................................. \( (4.4) \)

ii) The cost of travel on each link other than link \( e \) will change due to the change in the number of vehicles using that link:

\[ c_{a}(f_{a}^{*}) = c_{a}(f_{a} + \delta f_{a}) \]  for \( a = e \)

\[ = c_{a}(f_{a}) + \delta f_{a} c'_{a}(f_{a}) \]

\[ = c_{a}(f_{a}) + \delta f_{a} g_{a} \]  ...............\( (4.5) \)

iii) The cost of travel on link \( e \) will be affected by two factors: the change in the number of vehicles using link \( e \) and the change \( \varepsilon_{e} \) in the cost function of travel on link \( e \).
\[ c_e(f_e) \text{ is replaced by } (c_e + \varepsilon_e)(f_e + \delta f_e) \]
\[ = c_e^*(f_e^*) + c_e^*(f_e) + \varepsilon_e + \delta f_e c_e^*(f_e) \]
\[ c_e^*(f_e^*) = c_e(f_e) + \varepsilon_e + g_e \delta f_e \]  \hspace{1cm} (4.6)

The new cost of travel on route \( p \), \( C_p(F) \), is given by \( \sum \delta a^p c_a(f_a^*) + \delta e^p c_e^*(f_e) \) or using equations (4.5) and (4.6)

\[ C_p(F^*) = \sum \delta a^p c_a(f_a^*) + \delta e^p c_e^*(f_e) + \varepsilon_e + g_e \delta f_e \]

\[ = \sum \delta a^p c_a(f_a) + \sum \delta a^p f_a g_a + \delta e^p \varepsilon_e \]

\[ = C_p(F) + \delta e^p \varepsilon_e + \sum \delta a^p g_a \sum \delta a^q \delta F^q \]

i.e.

\[ C_p(F^*) = C_p(F) + \delta e^p \varepsilon_e + \sum \delta F^q \sum \delta a^q \delta a^p g_a \]

In matrix form

\[ C(F^*) = C(F) + \xi \varepsilon_e + A^{tr} G A \Delta F \]  \hspace{1cm} (4.7)

where \( G \) is a \( k \times k \) diagonal matrix whose \((a,a)\) entry is \( g_a \),

and \( \xi \) is a \( Q \times 1 \) vector, the transpose of row \( e \) of matrix \( A \).

Therefore \( B^{tr} \Delta U_w - \xi \varepsilon_e = A^{tr} G A \Delta F \)  \hspace{1cm} (4.8)

and since \( d \) is fixed, equations (4.2) and (4.8) give

\[ \begin{pmatrix} A^{tr} G A & -B^{tr} \\ B & 0 \end{pmatrix} \begin{pmatrix} \Delta F \\ \Delta U_w \end{pmatrix} = \begin{pmatrix} -\xi \\ 0 \end{pmatrix} \varepsilon_e \]  \hspace{1cm} (4.9)

[If \( d_{w_i} \) is a function of the cost \( U_{w_i} \), this may be taken into account here].

The left hand side is identical to that of equations (9-11) in Dafermos & Nagurney (1984).
Assuming we have Q used routes, we have Q + Z equations (4.2) and (4.8) with Q + Z unknowns, so that we can in general determine changes in $\Delta F$ and $\Delta U^w$ created by a change $\varepsilon_e$ in the cost of link e.

Hence,

$$\delta U^w_1 = \begin{vmatrix}
A^r G A - B^r_1 & B^r  \\
B & 0
\end{vmatrix} \varepsilon_e = \frac{\Delta^e_1}{\Delta} \varepsilon_e$$ \hfill (4.10)

where $B^r_1$ denotes matrix B with the ith row replaced by $\xi^r$,

and $\delta F^i_e$ is the change in flow on route i due to a change in the cost of travel of link e, and $\Gamma^i$ is the determinant of a $(Q \times Q)$ matrix analogous to matrix

$$\left(\begin{array}{cc}
A^r G A & - B^r  \\
B & 0
\end{array}\right)$$

with column i replaced by $(-\xi^r)$

The denominator of equations (4.10) and (4.11) is equivalent to the numerator $D^i$ derived in Steinberg and Zangwill (1983), and the denominator in Dafermos and Nagurney (1984) and was proved positive by the latter.

We shall present a graph-theoretic approach to the evaluation of the determinants of equations (4.10) and (4.11), again proving the denominator positive and attempting to predict the sign of the numerators.
4.4 Reducing The Size Of The Network.

Suppose that the set of links \( L \) is split into a congested set \( L_1 \) with \( g_a > 0 \) and an uncongested set \( \overline{L}_1 \) with \( g_a \) effectively zero. Assume that \( L_1 \) contains \( r \) links and \( \overline{L}_1 \) contains \( k-r \) links, then re-arrange matrix \( A \) and matrix \( G \) so that

\[
A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} \quad \text{and} \quad G = \begin{pmatrix} G_1 & 0 \\ 0 & 0 \end{pmatrix}
\]

where \( A_1 \) and \( A_2 \) are the link-path incidence matrices for the \( r \) congested and \( k-r \) uncongested links respectively, and \( G_1 \) is a \( r \times r \) diagonal matrix.

Then the analogues to equations (4.7), (4.9) and (4.10) are:

\[
C(F^*) = C(F) + \xi \epsilon_e + A_1^{tr} G_1 A_1 \Delta F \quad \text{............... (4.12)}
\]

and

\[
\begin{pmatrix} A_1^{tr} G_1 A_1 - B^{tr} \\ B \end{pmatrix} \begin{pmatrix} \Delta F \\ \Delta U_w \end{pmatrix} = \begin{pmatrix} -\xi \\ 0 \end{pmatrix} \epsilon_e \quad \text{... (4.13)}
\]

so that

\[
\delta U_{w_i} = \frac{\Delta w_i}{\Delta} = \left| \begin{array}{cc} A_1^{tr} G_1 A_1 & - B^{tr} \\ B & 0 \end{array} \right| \epsilon_e \quad \text{... (4.14)}
\]

and the analogue to equation (4.11) is

\[
\delta F_i = \frac{\Gamma_i^{(1)}}{\epsilon_e} \quad \text{... (4.15)}
\]

where \( \Gamma_i^{(1)} \) is the determinant of a \((Q \times Q)\) matrix analogous to matrix \( \begin{pmatrix} A_1^{tr} G_1 A_1 & - B^{tr} \\ B & 0 \end{pmatrix} \) with column \( i \) replaced by \(- (\xi^{tr}_{i}) \).
If the set $L_1$ does not contain a cycle then matrix $A_1$ represents the link-path incident matrix for a new graph $N$ formed by removing all links $aeL_1$ and amalgamating the nodes on either side of $a$, without creating loops.

If link $e$ is uncongested it is advisable to keep it in the new graph $N$ as this will help in determining the sign of the terms in $\Delta^e_{w_1}$.

However if $L_1$ contains a cycle of uncongested links, then the determinants of equation (4.14) are zero, and the solution of the assignment problem is not uniquely determined. In that case:

(i) If link $e$ is not part of that cycle, then equations (4.13) and (4.14) will hold, but will contain redundant conditions, which must be removed from equation (4.13).

(ii) If link $e$ is on the cycle, then the change in flow $\delta f_e$ need not be small and the linearization fails. This is because the original solution to the assignment problem is not uniquely determined, and the redundant equations can not all be removed.

4.5 The Single-Origin/Destination Network.

For a single-origin single-destination network $N(V,L)$, $Z = 1$, and the maximum number of independent cycles is $k-v+1$. By applying suitable row subtraction operations and corresponding column operations to $\Delta$ (i.e. if we subtract row $i$ from row $j$ then column $i$ must be subtracted from column $j$), we obtain a determinant $M^{r\text{CM}}$ of order $Q-Z = k-v+1$. $M^{r\text{tr}}$ is the cycle-link incidence matrix whose $(a, cy)$ entry is:

\[
\begin{cases}
1 & \text{if link } a \text{ belongs to cycle } cy \text{ and travels in the same direction.} \\
-1 & \text{if link } a \text{ belongs to cycle } cy \text{ and travels in the opposite direction.} \\
0 & \text{otherwise (link } a \text{ does not belong to cycle } cy).
\end{cases}
\]

see example on page 52.
The following definitions from Seshu & Reed (1961) are stated here for convenience.

**Elementary vector**

The vector \( \mathbf{R}_1 \) of the linear space whose elements are vectors with real integer components is elementary if it is nonzero and there is no other vector \( \mathbf{R}_2 \) in the space which has nonzero elements only at a proper subset of the positions in which \( \mathbf{R}_1 \) has nonzero elements (def. 5 - 12).

**Primitive vector.**

A vector \( \mathbf{R} \) of the linear vector space defined by the rows of \( \mathbf{F}_d \) (of real integral elements) is primitive if it is elementary and all of its entries are 1, -1 or 0 (def. 5 - 13).

**Real) regular matrix.**

The matrix \( \mathbf{F}_d \) of real integral elements is regular if to every elementary vector in the linear vector space spanned by the rows of \( \mathbf{F}_d \), there corresponds a primitive vector in the linear vector space with nonzero entries in the same position (def. 5 - 14).

**Fundamental cycle.**

The fundamental cycle of a connected graph \( N \) for a tree \( T \) and a link \( a \) (not belonging to \( T \)), is the cycle formed by link \( a \) and its unique tree route (def. 2 - 8).

**E - Matrix.**

A matrix \( \mathbf{F}_d \) of real elements is an E - matrix if the determinant of every square submatrix of \( \mathbf{F}_d \) is \( \pm 1 \) or 0.

**Major submatrix.**

A major submatrix of matrix \( \mathbf{F}_d(m \times n) \) is any \((m \times m)\) submatrix of \( \mathbf{F}_d \) where \( m \leq n \).

**A matrix in normal form.**

A matrix \( \mathbf{F}_d \) is in normal form if it has a major submatrix which is a unit matrix.
$M^{tr}$ is a regular matrix and it is always possible to transform it, by row subtraction operations into normal form, say $\tilde{M}^{tr}$ (basic theorem 4H, Rockafellar 1984; also lemma 5-25a, Seshu & Reed 1961). $\tilde{M}^{tr}$ has rank $k-v+1$. Hence it is a fundamental matrix, and by theorem 5-25 of Seshu & Reed (1961), "Every regular matrix $F_d$ in normal form is an E-matrix". Every square submatrix of $\tilde{M}^{tr}$ has a determinant equal to $\pm 1$ or $0$, and therefore the determinant of any major square submatrix of $M^{tr}$ is $\pm 1$ or $0$.

The following two theorems are only slight modifications of standard results in circuit theory (Seshu & Reed, chapter 7, 1961) (Moon, chapter 3, 1970), (Percival 1953), (Tutte, chapter 6, 1984); they are given here for convenience. This is for the single origin-destination network - so $B$ is a matrix with only one row.

**Theorem 4.5.1**

For a graph $N(V,L)$

$$\Delta = \begin{vmatrix} A^{tr}GA - B^{tr} \end{vmatrix} = \sum_{T} \prod_{T} G_{a}$$

where $T$ is the complement of the spanning tree $T$ of the undirected graph $N^*$, and the summation is over all such spanning trees.

**Proof**

The matrix $M^{tr}G$ is a $(Q-Z)\times(k)$ matrix where each column $a$ of $M^{tr}G$ is equal to row $a$ of $M$ multiplied by $g_a$. According to the Binet-Cauchy theorem

$$|M^{tr}GM| = \sum |I_s| |R_s|$$

where the summation is over all $(Q-Z) \times (Q-Z)$ submatrices $I_s$ of $M^{tr}G$ and the corresponding submatrices $R_s$ of $M$. Therefore $|M^{tr}GM| = \sum |R_s| \prod_{T} G_{a}$ where the product is over all links $a$ represented by the rows of $R_s$. However
$R_s$ is non-singular if and only if the rows of $R_s$ correspond to the complement of a spanning tree of $N^*$, (Theorem 4-11, Seshu and Reed 1961), where $R_s$ is a major square submatrix of $M$.

Hence

$$|R_s|^2 = \begin{cases} 1 & \text{if the set of links represented by the rows of } R_s \text{ are a complement of a spanning tree } T \text{ of } N^*, \\ 0 & \text{otherwise}, \end{cases}$$

and $|M^{tr}G_M| = \sum_{T \in T^*} g_a$, where $T$ is a spanning tree of $N$ and $T$ is its complement.

**Theorem 4.5.2**

For a graph $N(V,L)$, the complement of the links forming the columns of $R_s$ is a tree of $N^*$.

$$\Delta_{wi} = \begin{vmatrix} A^{tr}G_A & B^{tr} \\ B & 0 \end{vmatrix} = \sum_{T \in T^*} \frac{1}{\pi} g_a$$

where the summation is over all spanning trees $T_e$ of $N^*$, such that $T_e$ contains a route (p) between the origin and the destination containing link $e$. The sign will be positive if the route (p) travels along link $e$ in the same direction and negative if it travels along link $e$ in the opposite direction.

**Proof**

Define a new graph $N_1(V,L_1)$ identical with $N$ except that

i) link $e$ is removed and,

ii) the origin and destination nodes are identified (combined).
By applying suitable row subtraction operations and corresponding column operations to $A^e$, we obtain a determinant $M^\text{tr} G M_1^*$ of order $Q-Z$ where:

- $G$ is the same diagonal matrix as in theorem 4.5.1.
- $M^\text{tr}$ is the same cycle-link matrix as in theorem 4.5.1.
- $M_1^*$ is the link-cycle incidence matrix for graph $N_1^*$.

Let $R_S$ be any $(Q-Z) \times (Q-Z)$ square submatrix of $M$, and $R_{1S}$ be any $(Q-Z) \times (Q-Z)$ square submatrix of $M_1$. $R_S$ and $R_{1S}$ are major square submatrices of $M$ and $M_1$ respectively. Hence,

- $A : |R_S| = 0$ if the complement of the links forming the columns of $R_S$ is not a tree of $N^*$,

- $B : |R_S| = \pm 1$ if the complement of the links forming the columns of $R_S$ is a tree of $N^*$,

if $|R_S| = 0$ then $|R_S| \cdot |R_{1S}| = 0$, so we are only interested in case $B$.

The links forming the columns of matrix $R_S$ in case $B$ above, are the complement of one of the following types of trees.

I - A tree that does not include link $e$.

II - A tree that includes link $e$, but is not part of the route that joins origin destination pair $w_i^e$.

III - A tree that includes link $e$ as part of the route that joins the origin to the destination.

In case I, link $e$ belongs to the columns of $R_S$ then the analogous row of $R_{1S}$ corresponding to link $e$ consists entirely of zeros, since link $e$ does not belong to graph $N_1^*$, and therefore $|R_{1S}| = 0$ and $|R_S| \cdot |R_{1S}| = 0$. Hence complements of trees of type I do not appear in $\Delta^e_{w_i}$. 
In case II, assume that route $q$ joins origin destination pair $w_i$. The links forming route $q$ will also belong to the columns forming $R_S$, and because the origin and destination of pair $w_i$ are identified in $N_1^*$, $q$ will form a cycle in $N_1^*$. Hence $|R_S| = 0$ and $|R_S| \cdot |R_{1S}| = 0$.

Therefore, the only time where $|R_S| \cdot |R_{1S}| \neq 0$ occurs in case III. In that case any route $p$ joining origin destination pair $w_i$ in $N$ will not form a cycle in $N_1^*$, because link $e$ which is part of route $p$ is missing from graph $N_1$.

Finally, the sign of $|R_S|$ is equal to the sign of $|R_{1S}|$ if route $p$ travels along link $e$ in the same direction, hence $|R_S| \cdot |R_{1S}| = 1$; and $|R_S| \cdot |R_{1S}| = -1$ if route $p$ travels along link $e$ in the opposite direction, since $A_{w_i}^e$ is equivalent to the complement of the k-linkage between the origin and the initial point of link $e$, and the terminal point of link $e$ and the destination (Percival 1953). This completes the proof of theorem 4.5.2.

As we shall see, the individual terms of the denominator determinant $\Delta$ play an important role in what follows. The elements of the determinant are sums of the $g_a$, so that when expanded $\Delta$ is the sum of products of the $g_a$. In the single origin-destination case $\Delta$ is given by theorem 4.5.1.

The following results are immediate consequences of the theorem, stated as corollaries for ease of reference. Given a product of the form $g_{a_1} g_{a_2} g_{a_3} \ldots g_{a_v}$, we shall call the (unordered) set of links $h_t = \{a_1, a_2, \ldots a_v\}$ the $v$-tuple related to the product, and call $\prod_{a_i \in h_t} g_{a_i}$ the product related to tuple $h_t$. 
Corollary 4.5.1

Every term appearing in $\Lambda$ or in $\Lambda^e_{wi}$ is the product related to a tuple of length $Q - Z = v$.

Let $H$ be the set of all such tuples $h_t$. $h_t$ relates to $\Pi g_a$, from theorem 1 and theorem 2 the product $\Pi g_a$ is over all links represented in the rows of $R_s$, which is of rank $Q - Z$. Therefore every $h_t$ is of length $Q - Z$.

Corollary 4.5.2

In the single-origin single-destination case, any particular product $\Pi g_a$ appears at most once in $\Lambda$ and at most once in $\Lambda^e_{wi}$.

Since $|R_s| = \pm 1$ or 0, then every product $\Pi g_a$ appears only once in $\Lambda$.

In the single O-D network the complement of every tuple $h_t$ of $\Pi g_a$ is a set of links with a route, $p$, joining the origin to the destination. $p$ travels every link only once, so $\Pi g_a$ will appear only once in every $\Lambda^e_{wi}$ where $e$ is a link used by route $p$.

Corollary 4.5.3

All products $\Pi g_a$ have a positive sign in $\Lambda$.

Since $|R_s|^2 = +1$ or 0 then every product $\Pi g_a$ will be added to $\Lambda$.

It should be noted that any term $\Pi g_a$ may be positive or negative in $\Lambda^e_{wi}$.
From the proof of theorem 4.5.2 the complement of \( \nu \)-tuple \( h_t \) of \( \Pi g_a \) is a set of links with a route \( p \) joining the origin to the destination. If route \( p \) travels along link \( e \) in a confluent or counterfluent direction then \( \Pi g_a \) will be added or subtracted respectively from \( \Delta_{w_i}^e \).

Corollary 4.5.4

None of the terms in \( \Delta_{w_i}^e \) contains \( g_e \).

From theorem 4.5.2, each term is \( \Pi g_a \). Each tree \( T_e \) contains \( e \) so no complement \( \overline{T_e} \) contains \( e \).

Example

Assume a single origin-destination network \( N(V,L) \) with seven links and five nodes. This network consists of four independent routes which are all in use (Fig. 4.2).

Route 1 contains links 1,7
Route 2 contains links 2,6,7
Route 3 contains links 3,4,6,7
Route 4 contains links 3,5

Here \( \delta U = \frac{\Delta e}{\Delta} \epsilon_e \) (Figure 4.2)
where

\[
\Delta_e = \begin{vmatrix}
g_1 + g_7 & g_7 & g_7 & 0 & -\delta_{e1} 
g_7 & g_2 + g_6 + g_7 & g_6 + g_7 & 0 & -\delta_{e2} 
g_7 & g_6 + g_7 & g_3 + g_4 + g_6 + g_7 & g_3 & -\delta_{e3} 
0 & 0 & g_3 & g_3 + g_5 & -\delta_{e4} 
1 & 1 & 1 & 1 & 0
\end{vmatrix}
\]

and

\[
\Delta = \begin{vmatrix}
g_1 + g_7 & g_7 & g_7 & 0 & -1 
g_7 & g_2 + g_6 + g_7 & g_6 + g_7 & 0 & -1 
g_7 & g_6 + g_7 & g_3 + g_4 + g_6 + g_7 & g_3 & -1 
0 & 0 & g_3 & g_3 + g_5 & -1 
1 & 1 & 1 & 1 & 0
\end{vmatrix}
\]

By carrying out the following subtraction operations on \( \Delta \):

row 1 - row 2, row 2 - row 3, row 3 - row 4, column 1 - column 2, column 2 - column 3, column 3 - column 4,

we get

\[
\Delta = \begin{vmatrix}
g_1 + g_2 + g_6 & -g_2 & -g_6 & 0 & 0 
g_2 & g_2 + g_3 + g_4 & -g_4 & -g_3 & 0 
g_6 & -g_4 & g_4 + g_5 + g_6 + g_7 & -g_5 & 0 
0 & -g_3 & -g_5 & g_3 + g_5 & -1 
0 & 0 & 0 & 1 & 0
\end{vmatrix}
\]

By expanding along the last column then the bottom row, we
obtain

\[ \Delta = \begin{vmatrix} M^{\text{tr}} G M \end{vmatrix} = \begin{vmatrix} g_1 + g_2 + g_6 & -g_2 & -g_6 \\ -g_2 & g_2 + g_3 + g_4 & -g_4 \\ -g_6 & -g_4 & g_4 + g_5 + g_6 + g_7 \end{vmatrix} \]

here

\[ M^{\text{tr}} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 1 & 1 \end{pmatrix} \]

on expansion, we find that

\[ \Delta = \sum_{i=1}^{4} g_i g_2 g_i + g_1 g_2 g_5 + g_1 g_2 g_6 + g_1 g_3 g_4 + g_1 g_3 g_5 + g_1 g_4 g_5 + g_1 g_5 g_6 + g_1 g_6 g_7 + g_1 g_7 g_8 + g_1 g_8 g_9 + g_1 g_9 g_10 \]

Similarly, we find that each term in \( \Delta \) is related to exactly one of the spanning trees of \( N^* \) once removed.
Consider the term $g_1g_2g_3$; if links 1, 2, and 5 are removed from $N^*$, links {3, 4, 6, 7} remain which forms a spanning tree of $N^*$ and contains route 3 travelling along link 4 in the same direction of link 4 in $N$, therefore $g_4g_5$ is added to $\Delta_4$.

Now consider the term $g_1g_3g_7$; if link 1, 3, and 7 are removed from $N^*$, the links {2, 4, 5, 6} remain, forming a spanning tree of $N^*$ containing route 2, −4, 5, which travels along link 4 in the opposite direction. Hence, the term $g_1g_3g_7$ is subtracted from $\Delta_4$.

4.6 The Multi-Origin/Destination network

In the multi-origin/destination network, when row $i$ is subtracted from row $j$ and likewise column $i$ from column $j$ in the matrix $\begin{pmatrix} A^{tr} & G^{tr} \\ B & 0 \end{pmatrix}$ the routes which are represented by rows $i$ and $j$ must belong to the same origin destination pair. The resulting matrix $M^{tr}G_M$ will then be of order Q−Z. The set of Q−Z cycles represented by the
The set of all routes joining origin destination pair \( w \) will consist of used and unused routes. The used routes will have an equal travel cost not greater than the cost of travel on the unused routes. We assume that every origin destination pair has a fixed flow, so a unit of flow between \( O \rightarrow D_1 \), and a unit of flow between \( O \rightarrow D_2 \) cannot be replaced by flows \( O \rightarrow D_2 \) and \( O \rightarrow D_1 \) as in the corresponding electrical network. Therefore it may be impossible to use all the cycles of graph \( N(V,L) \) in rearranging the flow.

It may be possible to find a new graph \( N^{*}(V,L) \), that contains the same number of links as graph \( N^{*}(V,L) \), but contains only "the cycles represented in \( A \)". If this is the case, the network developed for the single origin destination pair may be used.

Example

Consider the graph \( N(V,L) \) with six links, four nodes and two origin destination pairs. This network consists of four independent routes which are all in use (Figure 4.3).

0 - D1: Route 1 = 1, 4
Route 2 = 3, 5
O - D2: Route 3 = 2, 3
Route 4 = 4, 6

Figure 4.3
The set of "cycles represented in $\Delta$" is
\[ c_1 = 1, -3, 4, -5 \]
\[ c_2 = 2, 3, -4, -6 \]
and the cycles that may appear in "the cycles represented in $\Delta$" are
\[ c_1 = 1, -3, 4, -5 \]
\[ c_2 = 2, 3, -4, -6 \]
\[ c_3 = c_1 + c_2 = 1, 2, -5, -6 \]

Whereas the set of cycles in graph $N(V, L)$ is
\[ c_1 = 1, -3, 4, -5 \]
\[ c_2 = 2, 3, -4, -6 \]
\[ c_3 = 1, 2 \]
\[ c_4 = 5, 6 \]

In this case we can create a new graph $NN(V_1, L)$, with six links, two origin destination pairs but only "the cycles represented in $\Delta$" (Fig 4.4).

Figure 4.4

$NN(V, L_1)$ represents the electrical network equivalent of the transportation network $N(V, L)$ and theorem 4.5.1 will apply to graph $NN$ in the same way it applies for a single origin-destination network.

Graph $NN^*$ does not always exist, and to explain the existence of graph $NN$ and the evaluation of $\Delta$ and $\Delta_w^e$, the following definitions are necessary.
Definition

The set of pseudo cycles (PC) of the network is the set of cycles created by the multiples of rows of \( M^{tr} \),

\[ \text{pc} = \sum_{i=1}^{\lambda} \lambda_i \text{cy}_i, \]

where \( \lambda_i \) is an integer, not all \( \lambda_i \) are zero, and \( \text{cy}_i \) is a cycle belonging to the "cycles represented in \( \Delta \)”, where PC satisfies the following conditions:-

i) No \( \text{pc}^* \in \text{PC} \) exists such that \( \text{pc} = a \text{pc}^* \) with \( |a| > 1 \)

ii) No \( \text{pc}^* \in \text{PC} \) exists with non-zero elements at a proper subset of the non-zero elements of \( \text{pc} \).

PC is essentially the set of non-zero cycles created from the "cycles represented in \( \Delta \)".

Definition

The set of pseudo-routes (PP) for origin destination pair \( w_i \) is the set of all routes created by the addition of one or more routes joining origin destination pair \( w_i \), and a pseudo cycle satisfying the following conditions:

i) If \( p_1 \in \text{PP} \), no route \( p_2 \in \text{PP} \) exists such that \( p_1 = \alpha p_2 \) where \( \alpha > 1 \)

ii) If \( p_1 \in \text{PP} \), no route \( p_2 \in \text{PP} \) exists such that non-zero elements of \( p_1 \) occur at a proper subset of the non-zero elements of route \( p_2 \).

Definition

Pseudo-Trees (PT):– For graph \( N(V,L) \), the set of links \( pt \) is a pseudo-tree of \( N(V,L) \) if the addition of any new link \( a \) creates a pseudo-cycle.
In the single origin destination network and the multi origin destination network where \( Q - Z = k-v+1 \), the matrix \( M \) is regular and the set of \( PC \) will simply be the set of all cycles of graph \( N \). In such case graph \( NN(V,L) \) will be graph \( N(V,L) \) itself. Theorems 4.5.1 and 4.5.2 will apply to such multi origin destination networks in the same way that they do for the single origin destination case. However if \( Q - Z \) is less than \( k-v+1 \), then the matrix \( M \) will either be:

1. Regular (as defined in section 4.5) or
2. Irregular

### 4.7 Real Regular Matrix \( M \).

#### 4.7.1 Change in the Average Cost of Travel.

If \( M \) is regular, then \( M \) has a normal form and the determinants of all major square submatrices of \( M \) equal \( \pm 1 \) or 0. Therefore all the tuples appear only once in the denominator (as in lemma 4.5.2). Lemma 5-25b of Seshu and Reed states that 'The rows of the regular matrix in normal form are primitive vectors': hence no cycle belonging to \( PC \) will have a repeated link. An example of a graph with a regular matrix \( M \) is the ladder network with single or two way traffic and multi origin destination; none of the pseudo-cycles of such graphs contain a repeated link (see Fig 4.5).

The problem of when a graph exists with a given link-cycle matrix is of long standing and an answer takes up 57 pages of 'Transactions of the American Mathematical Society' (Tutte, 1958 & 1959).

![Figure 4.5](image-url)
In the above case graph \( NN \) exists if no normal form of \( M \) contains a cut set matrix of either of the two basic nonplanar graphs of Kuratowski (Kuratowski 1930, Fig 4.6 a,b) [Theorem 5-29, Seshu and Reed].

![Figure: 4.6a](image1.png) ![Figure: 4.6b](image2.png)

All the terms appearing in \( \Delta \) are complements of trees of \( NN \). Finding the graph \( NN(V,L) \) is not necessary to determine \( \Delta \) and is mentioned here only to justify the use of terms such as pseudo-routes and pseudo-trees, and also to show the similarities and differences between the transportation network and the electrical network.

To find the set of complements of trees as in \( \Delta \), the following method can be used when the matrix \( M \) is regular.

**Method 1.**

a) List all \( v \) cycles "represented in \( \Delta \)."

b) Choose one link from each cycle to form an \( v \)-tuple \((l_1 \ldots l_v)\); where \( H \) is the set of all such \( v \)-tuples.

c) Remove from \( H \) any \( v \)-tuple containing a repeated link.

d) Remove from \( H \) any \( v \)-tuple which occur an even number of times.

e) Finally, if any \( v \)-tuple occurs an odd number of times in \( H \), remove all but one occurrence. Let \( H \) be the set of all remaining \( v \)-tuples.

The above method is programmed in FORTRAN on the VAX 11/780. Programme listing is available in the appendix.
Example.

Consider the network drawn earlier in Figure 4.2. The cycles represented in $\Delta$ are:

- $c_y_1 = 1, 2, 6$
- $c_y_2 = 2, 3, 4$
- $c_y_3 = 4, 5, 6, 7$

Hence $H = \{124, 125, 126, 127, 134, 135, 136, 137, 144, 145, 146, 147, 224, 225, 226, 227, 234, 235, 236, 237, 244, 245, 246, 247, 246, 247, 256, 256, 257, 346, 356, 366, 367, 446, 456, 466, 467\}$ less terms with repeated links less tuples repeated an even number of times.

Therefore


Hence $\Delta = g_1g_2g_4 + g_1g_2g_5 + g_1g_2g_6 + g_2g_3g_7 + g_1g_2g_4 + g_1g_2g_5 + g_1g_2g_6 + g_1g_2g_7 + g_2g_3g_4 + g_2g_3g_5 + g_2g_3g_6 + g_2g_3g_7 + g_2g_4g_5 + g_2g_4g_6 + g_2g_4g_7 + g_2g_5g_6 + g_2g_5g_7 + g_2g_6g_7 + g_2g_5g_7 + g_2g_6g_7 + g_2g_6g_7$ as shown earlier.

To justify steps (d) and (e) above, let $\Phi H$ be the set of all $v$-tuples produced by steps (a), (b) and (c). Every tuple $h_t$ in $\Phi H$ is repeated $\alpha_t$ times where $\alpha_t$ is an integer greater than zero.

Let $R_h$ be a major square submatrix of $M$, where the columns of $R_h$ are links that constitute tuple $h_t$.

Theorem 4.7.1.

For any network $N(V, L)$, the determinant of $R_{ht}$ equals $\alpha_t$ modulo 2.
Proof

For every term $g_1 \ldots g_v$ produced by choosing one link at a time from each of the sets of pseudo cycles, as in $\Phi$, let $R_{h_t}$ be the $v \times v$ matrix

$$
R_{h_t} = \begin{pmatrix}
   a_{11} & a_{12} & \cdots & a_{1v} \\
   . & . & & . \\
   a_{v1} & a_{v2} & \cdots & a_{vv}
\end{pmatrix}.
$$

The number of times the term $g_1 \ldots g_v$ appears in $\Phi$ is

$$
\sum_{\pi} |a_{11} \ldots a_{v1}| |a_{12} \ldots a_{v2}| \ldots |a_{1v} \ldots a_{vv}| (4.16)
$$

where the sum is over all permutations $\pi$ on $v$ objects.

The determinant of $R_{h_t}$, $|R_{h_t}|$, equals

$$
\sum_{\pi} (\text{sign } \pi)(a_{11}\pi(1) \ldots a_{v1}\pi(v)) \ldots (4.17)
$$

where $\text{sign } \pi$ is either $+$ or $-$ depending on whether the permutation is even or odd.

The difference between equation (4.16) and (4.17) is

$$
\sum_{\pi} \{ |(\text{sign } \pi)(a_{11} \ldots a_{v1}\pi(v))| - (\text{sign } \pi)(a_{11} \ldots a_{v1}\pi(v)) \}
$$

If any of the $a_{ji}(j)$ terms is zero, then both terms in the RHS of equation (4.18) are zero. Otherwise if $\text{sign } \pi = \pm 1$ then

$$
\text{RHS of expression (4.18)} = \begin{cases}
   \{ | -1 | - (-1) \}(a_{11}\pi(1) \cdots a_{v1}\pi(v)) \\
   \{ | 1 | - (1) \}(a_{11}\pi(1) \cdots a_{v1}\pi(v))
\end{cases}
$$

$$
= \{ 0 \text{ or } 2 \}(a_{11}\pi(1) \cdots a_{v1}\pi(v)).
$$

Therefore, the number of times $(g_1 \ldots g_v)$ appears in the determinant of $M$ is $a_i$ modulo 2.
For a real regular matrix M, \(| R_{t} | = \pm 1\) or 0. Then an even \(a_{t}\) implies \(| R_{t} | = 0\), and so such a \(v\)-tuple \(h_{t}\) is a cut set and should be discarded. Likewise an odd \(a_{t}\) implies \(| R_{t} | = \pm 1\). So \(| R_{t} |^2 = 1\) and hence \(h_{t}\) should appear only once in \(\Delta\).

To evaluate \(\Delta^{e}_{\omega_{1}}\), the set of all pseudo-routes for the O-D pair, \(PP^{\omega_{1}}\), must be found. This can be achieved by using the following method.

a) List the set of cycles \(CY\).

b) List the set of used routes \(PR_{\omega_{1}}\), include these in \(PP_{\omega_{1}}\).

c) For every route \(p \in PP_{\omega_{1}}\) and cycle \(cy\) that have a link in common, add or subtract the cycle from the route to produce a putative route \(p^{*}\).

d) If any subset of \(p^{*}\) belongs to \(PP_{\omega_{1}}\), discard \(p^{*}\).

\(\text{otherwise add } p^{*} \text{ to } PP_{\omega_{1}}.\)

e) If \(p^{*}\) is a subset of some route \(p \in PP_{\omega_{1}}\), discard \(p\).

f) Repeat the process until all routes in \(PP_{\omega_{1}}\) are exhausted.

The above method is programmed in FORTRAN on the VAX 11/780. Programme listing and a flow chart are available in the appendix.

For any product related to a tuple \(h_{t}\) that appears in \(\Delta\), if the complement of \(h_{t}\) contains only one pseudo-route, \(p_{i}\), then the product of \(h_{t}\) will appear in \(\Delta^{e}_{\omega_{1}}\), where \(e\) is a link belonging to the pseudo-route \(p_{i}\). If the complement of \(h_{t}\) contains more than one pseudo-route, then \(h_{t}\) is discarded. The product of \(h_{t}\) will be added to \(\Delta^{e}_{\omega_{1}}\) if \(p_{i}\) travels along link \(e\) in the same direction, and subtracted if \(p_{i}\) travels along link \(e\) in the opposite direction.

If the complement of tuple \(h_{t}\) is a pseudo-tree that contains a pseudo-route \(p_{j}\), where \(p_{j}\) travels along
link $e$ $\beta_j$ times, then the product related to tuple $h_t$ will appear $\beta_j$ times in $\Lambda^e_{w_i}$.

4.7.2 Flow Redistribution.

Let us now consider the effect of a change in the cost of travel on link $e$, on the flow on each link.

Let $\delta f_a^e$ be the change in the flow on link $a$ due to a change in the cost of travel on link $e$.

Then $\delta f_a^e = \sum \delta a_i \delta F_i^e$,

where $\delta F_i^e$ is the change in the flow on route $i$ due to the change in the cost of travel on link $e$.

From equation (4.11)

$$\delta F_i^e = \frac{\Gamma_i}{\Delta} \varepsilon_e,$$

where $\Gamma_i$ is the determinant of a $(Q \times Q)$ matrix analogous to matrix

$$\begin{pmatrix}
A_{tr} & -B_{tr} \\
B & 0
\end{pmatrix}$$

with column $i$ replaced by

$$\begin{pmatrix}
\varepsilon^tr \\
0
\end{pmatrix}.$$.

Therefore

$$\delta f_a^e = \sum \delta a_i \frac{\Gamma_i}{\Delta} \varepsilon_e$$

$$= \frac{\varepsilon_e}{\Delta} \Gamma_{ea}$$

where

$$\Gamma_{ea} = \sum \delta a_i \Gamma_i.$$

In particular

$$\Gamma_{ee} = \sum \delta e_i \Gamma_i.$$

First, we establish a surprisingly simple relationship between $\Gamma_{ee}$ and $\Delta$. For any $v$-tuple $h_v$, which contains link $e$, define the $v$-tuple $J^e_t$ by:
The determinant $\Delta = \sum_{h_t} \text{product related to tuple } h_t$

$$= \sum_{h_t} \prod_{a \in h_t} g_a$$

$$= \sum_{h_t} g_e \prod_{a \in h_t} g_a + \sum_{h_k} \prod_{a \in h_k} g_a,$$

where $h_k$ runs over the tuples that do not contain link $e$. In what follows we assume that the cost of travel on link $e$ is increased.

**Theorem 4.7.2.**

The determinant $\Gamma_{ee}$ is precisely minus the coefficient of $g_e$ in the expansion of the denominator determinant $\Delta$; that is,

$$\Gamma_{ee} = -\sum_{h_t \in A^t} \prod_{a \in h_t} g_a$$

where the summation is over all tuples $h_t$ of $A$ that contain link $e$.

**Proof**

Consider the case where the cost of link $e$ is increased.

$$\Gamma_{ee} = \sum_{i} \delta_{ei} \Gamma_i$$

where $\Gamma_i$ is the determinant of matrix

$$\begin{pmatrix}
A_{tr} & B_{tr} \\
0 & 0
\end{pmatrix}$$

with column $i$ replaced by $-\xi_{tr}$.
Construct a \((Q+1)\times (Q+1)\) matrix with entries \(Y_{ij}\) as follows:

\[
Y_{ii} = -1
\]

\[
Y_{ij} = \begin{cases} g_e & \text{if link } e \text{ belongs to route } j-1 \\ 0 & \text{otherwise} \end{cases}
\]

for \(j = 2, \ldots, Q-Z+1\)

\[
Y_{j1} = \begin{cases} -\delta_{a(j-1)} & \text{for } j = 2, \ldots, Q-Z+1 \\ 0 & \text{for } j > Q-Z+1 \end{cases}
\]

\[
Y_{ij} = m_{i-1,j-1} \text{ for } i, j = 2, \ldots, Q-Z+1
\]

where \(m_{i-1,j-1}\) is entry \((i-1,j-1)\) in

\[
\begin{pmatrix}
\text{A}^{tr}GA & -B^{tr} \\
B & 0
\end{pmatrix}
\]

Let \(\bar{\Gamma}\) be the determinant of the above matrix; then

\[
\bar{\Gamma} = \begin{vmatrix}
-1 & g_e \delta_{e1} & g_e \delta_{e2} & \cdots & g_e \delta_{e(Q-Z)} & 0 & 0 \\
-\delta_{e1} & A^{tr}GA & \cdots & \cdots & \cdots & -B^{tr} \\
-\delta_{e2} & 0 & A^{tr}GA & \cdots & \cdots & \cdots & -B^{tr} \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
-\delta_{e(Q-Z)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\end{vmatrix}
\]

From expanding the above determinant by the top row

\[
\bar{\Gamma} = -\left[ A + \sum g_e \Gamma_1 \delta_{e1} \right] 
\]

Alternately by subtracting row 1 from the rows with entry \(-1\) in column one for matrix \(\bar{\Gamma}\), and expanding the determinant along column one we get:

\[
\bar{\Gamma} = -\begin{vmatrix}
A^{tr}GA & -B^{tr} \\
B & 0
\end{vmatrix}
\]

where \(C\) is matrix \(G\) with each entry \(g_e\) replaced by zero.

\(\bar{\Gamma}\) is then a sum of products of the \(g_a\), but \(\bar{\Gamma}\) does not contain any terms in \(g_e\).
But $\Gamma_{ee} = \sum_{i} \delta_{ei} \Gamma_i$, so $\Delta = -g_{ee} \Gamma_{ee} - \Gamma$ from (4.19).

As $\Gamma$ is independent of $g_{ee}$, $\Gamma_{ee}$ is just the coefficient of $g_{ee}$ in the expansion of $\Delta$.

So $\Gamma_{ee}$ is the set of all tuple products of $j^e_t$:

$$\Gamma_{ee} = -\sum_{h^e_t \in \Gamma} g^a$$

Note that $\Gamma_{ee}$ is always negative; note also that this result is true whether $M$ is regular or irregular.

Theorem 4.7.3.

If the complement of $v$-tuple $h^e_t$ is a pseudo-tree, then the complement of $j^e_t$ contains a unique pseudo cycle travelling along link $e$.

Proof

Assume that the complement of $j^e_t$ contains two pseudo-cycles $c_y^i$ and $c_y^j$, such that:

$c_y^i$ travels along link $e$ $\alpha_i$ times
$c_y^j$ travels along link $e$ $\alpha_j$ times.

A new pseudo cycle $c_y^k$ can be formed from $c_y^i$ and $c_y^j$ as follows:

$c_y^k = \alpha_j c_y^i - \alpha_i c_y^j$

If $c_y^k$ is a zero vector, then pseudo cycle $c_y^i$ is a multiple of pseudo-cycle $c_y^j$, and hence only one will be included in the set of pseudo-cycles.

However, if $c_y^k$ is not a zero vector, then $c_y^k$ will have zero entries for at least link $e$ and all other links in common with $j^e_t$; pseudo-cycle $c_y^k$ does not contain link $e$ nor any of the links in $j^e_t$.

Hence the complement of $v$-tuple $h^e_t$ contains cycle $c_y^k$: contradiction, therefore either $c_y^i$ or $c_y^j$ are included in the set of pseudo cycles, and the complement of $j^e_t$ contains a unique cycle travelling along link $e$. 
From the flow conservation law, the total change in flow is zero, and therefore any diversion of flow must follow a closed circuit. Therefore the product of tuple \( j^e_t \) which appears in \( \Gamma_{ea} \) will also appear in \( \Gamma_{ee} \), where \( a \) is a link belonging to the pseudo cycle \( c_{y_i} \). That is,
\[
\Gamma_{ea} = \sum \pm \prod_{a \in j_t} g_a
\]
where the sum is over all tuples \( h_t \), such that the complement of contains a pseudo cycle passing through links \( a \) and \( e \).

The sign of the tuple product will be negative if links \( a \) and \( e \) travel in the same direction in \( c_{y_i} \), and positive if links \( a \) and \( e \) travel in the opposite direction in \( c_{y_i} \).

It seems useful to obtain the set of pseudo-cycles of a network; by a slight modification, the program used in the previous section to determine the set of pseudo-routes can be used to obtain the set of pseudo-cycles. One simply exchanges \( p \) and \( PP_{w_i} \) with \( c_{y_i} \) and \( CY \) in steps b, c and d. Note that \( g_e \) and \( g_a \) do not appear among the terms in \( \Gamma_{ae} \) or \( \Gamma_{ea} \).

**Lemma 4.7.1**

The product sum
\[
\Gamma_{ea} = \Gamma_{ae}
\]

**Proof**

For any tuple \( j^e_t \), where the product related to \( j^e_t \) appears in \( \Gamma_{ea} \), the complement of \( j^e_t \) contains a pseudo-cycle \( c_{y_i} \) that travels along links \( e \) and \( a \), therefore the product related to \( j^e_t \) will also appear in \( \Gamma_{ae} \).
If \( j^e_t \) is a tuple where the product related to \( j^e_t \) does not appear in \( r_{ea} \), then the complement of \( j^e_t \) does not contain a cycle travelling along links \( a \) and \( e \), therefore the complement of \( j^e_t \) does not appear in \( \Gamma_{ae} \) either.

If the product related to \( j^e_t \) is negative (positive), then links \( e \) and \( a \) travel in the same (opposite) direction in \( cy_1 \), therefore the product related to \( j^e_t \) will also be negative (positive) in \( \Gamma_{ae} \).

Note that theorems 4.7.2, 4.7.3 and Lemma 4.7.1 are true regardless of the type of matrix \( M \).

4.8 Irregular Matrix \( M \).

For an irregular matrix \( M \), the rows of \( M \) are not necessarily primitive, so the pseudo-cycles may contain repeated links. The determinant \(| R_{ht} |\) of any major square submatrix of \( M \), is not necessarily equal to \( \pm 1 \) or 0; so the product of any \( v \)-tuple \( h_t \) may appear more than once in the denominator \( \Delta \), but may appear \(| R_{ht} |^2 \) times; 4, 9, 16 ... times.

Example

Assume a two origin-destination network with eight links, four used routes and six nodes (Fig 4.7)

![Figure 4.7](image-url)
The used routes are:

For 0 – D 1

\[ P_1 = \text{links } 1, 2, 3 \]
\[ P_2 = \text{links } 4, 5, 6. \]

For 0 – D 2

\[ P_3 = \text{links } 2, 7, 5 \]
\[ P_4 = \text{link } 8. \]

So

\[
\Delta = \begin{array}{ccc}
g_1 + g_2 + g_3 & 0 & g_2 \ 0 & g_4 + g_5 + g_6 & g_5 \ 0 & 0 & -1 \end{array}
\]

And the set of pseudo cycles is

\[ c_y^1 = \text{links } 1, 2, 3, -4, -5, -6 \]
\[ c_y^2 = \text{links } 2, 5, 7, -8 \]
\[ c_y^3 = \text{links } 1, 2, 3, -4, -6, 7, -8 \]
\[ c_y^4 = \text{links } 1, 3, -4, -5, -6, -7, 8 \]

Note that \( c_y^3 \) and \( c_y^4 \) contain repeated links.

4.8.1 Flow Redistribution.

First, consider the effect of an increase in the cost of travel of a link on the flow fluctuation on other links in the network. Theorems 4.7.2, 4.7.3 and lemma 4.7.1 are still true, but now the pseudo-cycles may contain repeated links and the denominator repeated tuples.

If the complement of \( J_t^a \), the product of which appear in \( \Gamma_{ab} \), creates a pseudo cycle that includes link \( a \) repeated \( a \) times, and link \( b \) repeated \( b \) times, then the product related to tuple \( h_t \) will
appear in $\Delta$ at least $a^2$ times, and the effect of the change in the cost of travel on link $a$, on the flow on link $b$, will have the product related to $j^a_t$ appearing $\beta/a$ times for each product related to $j^a_t$ in $\Delta$. Hence, the total number of times the product related to $j^a_t$ appears in $\Gamma_{ab}$ is at least $a^2\beta/a = a\beta$ times. Likewise, if the complement of $j^a_t$ includes a pseudo cycle $p_c$, with link $a$ repeated $a_i$ times, and assuming that the product related to $h_t$ is repeated $a_i^2$ times in $\Delta$, then the product related to $j^a_t$ will appear $a_i^2$ times in $\Gamma_{aa}$. An upper bound on the number of times a tuple is repeated in $\Delta$ can be obtained from the square of the number of times $h_t$ is repeated in $P\Phi$ (the selection process, theorem 4.7.1).

Example

For the network in Fig. 4.7, the effect of an increase in the cost of travel on link 2 on the flow on link 1, $\Gamma_{21}$, is found as follows. Take the set of tuples $j^2_t$ leaving pseudo cycle

<table>
<thead>
<tr>
<th>$h_t$</th>
<th>$j^2_t$</th>
<th>leaving pseudo cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,5</td>
<td>5</td>
<td>1,2,2,3,-4,-6,7,-8</td>
</tr>
<tr>
<td>2,7</td>
<td>7</td>
<td>1,2,3,-4,-5,-6</td>
</tr>
<tr>
<td>2,8</td>
<td>8</td>
<td>1,2,3,-4,-5,-6</td>
</tr>
</tbody>
</table>

the other tuples including link 2 are rejected because the complement of their $j^2_t$ does not contain a unique pseudo cycle passing through links 1 and 2.

Link 2 appears twice in pseudo-cycle 1,2,2,3,-4,-6,7,-8, while link 1 appears only once in the same pseudo-cycle.

Hence, for every tuple product $g_2g_5$ appearing in $\Delta$, tuple product $g_5$ will appear $1/2$ times in $\Gamma_{21}$. So the total number of times tuple product $g_5$ appears in

$$\Gamma_{21} = \text{number of times } g_2g_5 \text{ is repeated in } \Delta \times 1/2$$

$$= 4 \times 1/2$$

$$= 2.$$ 

Likewise, $g_7$ and $g_8$ will appear once in $\Gamma_{21}$. The sign of $g_5$, $g_7$ and $g_8$ will be negative; links 1 and 2 travel in the same direction in the pseudo-cycles above.
Therefore \( \Gamma_{21} = -2g_5 - g_7 - g_8 \)

To check this:

<table>
<thead>
<tr>
<th>-1 0 ( g_2 ) 0 -1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 ( g_4 ) + ( g_5 ) + ( g_6 ) ( g_5 ) 0 -1 0</td>
</tr>
<tr>
<td>-1 ( g_5 ) ( g_2+ g_7 ) 0 0 -1</td>
</tr>
<tr>
<td>0 0 0 ( g_8 ) 0 -1</td>
</tr>
<tr>
<td>0 1 0 0 0 0</td>
</tr>
<tr>
<td>0 0 1 1 0 0</td>
</tr>
</tbody>
</table>

\[ \Gamma_{21} = -1 \begin{bmatrix} \( g_2+ g_5 \) \( -g_5 \) 0 \\ \( g_2+ g_5+ g_7 \) -\( g_7 \) -\( g_8 \) \\ 1 1 1 \end{bmatrix} = -2g_5 - g_7 - g_8 \]

Similarly

\( \Gamma_{22} = -(g_1+ g_3+ g_4+ g_6+ g_7+ g_8+ 4g_5) \)

\( \Gamma_{23} = -(2g_5+ g_7+ g_8) \)

\( \Gamma_{24} = 2g_5+ g_7+ g_8 \)

\( \Gamma_{25} = -(g_1+ g_3+ g_4+ g_6)+ g_7+ g_8 \)

\( \Gamma_{26} = 2g_5+ g_7+ g_8 \)

\( \Gamma_{27} = -(2g_5+ g_1+ g_3+ g_4+ g_6) \)

\( \Gamma_{28} = 2g_5+ g_1+ g_3+ g_4+ g_6 \)

Note the \( g_2g_5 \) appears four times in \( \Delta \), and the complement of link 5 contains the pseudo-cycle 1, 2, 3, 4, -5, -6, -7, -8.

Therefore \( g_5 \) will appear:

- 2 times in \( \Gamma_{21} \), with a negative sign
- 4 times in \( \Gamma_{22} \), with a negative sign
- 2 times in \( \Gamma_{23} \), with a negative sign
- 2 times in \( \Gamma_{24} \), with a positive sign
- 2 times in \( \Gamma_{26} \), with a positive sign
- 2 times in \( \Gamma_{27} \), with a negative sign
- 2 times in \( \Gamma_{28} \), with a positive sign

As in Figure 4.9.
The pseudo-cycle $1,2,2,3,-4,-6,7,-8$ is the combination of cycles $1,2,3,-4,-5,-6$ and $2,5,7,-8$. $g_5$ appears once for each of the links in cycles $1,2,3,-4,-5,-6$ and $2,5,7,-8$, causing the above mentioned order, see figure 4.8.

Likewise $g_2g_7$ appears only once in $\Delta$, and the complement of link 7 contains pseudo-cycle $1,2,3,-4,-5,-6$. Therefore $g_7$ will appear:

- Once in $\Gamma_{21}$, with a negative sign
- Once in $\Gamma_{22}$, with a negative sign
- Once in $\Gamma_{23}$, with a negative sign
- Once in $\Gamma_{24}$, with a positive sign
- Once in $\Gamma_{25}$, with a positive sign
- Once in $\Gamma_{26}$, with a positive sign

As in Figure 4.9

4.8.2 Change in The Average Cost of Travel.

Now, we shall consider the change in the average cost of travel between any origin destination pair, due to a change in the cost of travel of a link.
For any tuple created by the selection process, whose related product is to be included in \( \Delta \), the tuple must be the complement of a pseudo-tree of graph \( N \). Hence, the complement of every tuple related to a product in \( \Delta \) should include a unique pseudo-route joining each origin-destination pair.

If matrix \( M \) is irregular, the set of pseudo-routes created by the combination of a route and a cycle will not represent all the pseudo-routes of the graph. Some pseudo-routes are created only by the combination of two or more routes. Therefore in the algorithm in section 4.7.1 for finding all pseudo-routes, step(c) should be revised to allow routes of the form \( p^* = \lambda p + \mu cy \) where \( \lambda \geq 1 \).

**Example**

Consider the network in Fig 4.7.

The set of used routes and cycles are:-

**Origin-destination pair 1.**

- \( P_1 = \) links 1,2,3  
  - \( cy_1 = 1,2,3,-4,-5,-6 \)
- \( P_2 = \) links 4,5,6  
  - \( cy_2 = 2,5,7,-8 \)

The pseudo-routes are:-

- \( P_1 = 1,2,3 \)
- \( P_2 = 4,5,6 \)
- \( P_3 = P_1 - cy_2 = 1,3,-5,-7,8 \)
- \( P_4 = P_2 - cy_2 = -2,4,6,-7,8 \)
- \( P_5 = P_1 + P_2 = P_1 + P_3 = 1,3,4,6,-7,8 \)

Note that no subset of \( P_5 \) constitutes a pseudo-route, although \( P_5 \) consists of two routes from origin to destination.

Let \( h_t \) be a tuple such that the complement of \( h_t \) is a pseudo-tree with a pseudo-route \( p^*_i \) joining origin destination pair \( i \), and suppose that \( p^*_i \) can only be created by the addition of two or more pseudo-routes from origin to destination. To find the minimum number of pseudo-routes needed to create \( p^*_i \) we need to find the smallest positive integer solution \( k \) from:-

\[
kp_j + \sum_{i} \lambda_i cy_i = PR \quad \cdots \cdots \cdots (4.20)
\]
where \( PR \) is a \((1 \times k)\) vector such that all the elements in common with \( h^t \) are zero, also \( k \) and \( \lambda_i \) are integers, and \( p_j \) is a route joining origin-destination pair \( w_i \).

Consider equation (4.20) for only the links represented in \( h^t \):
\[
kp_j + \lambda R_{h^t} = 0
\]
therefore
\[
\lambda = \frac{p_j}{R_{h^t} - k}
\]
so
\[
X = \frac{(\text{adj } R_{h^t})p_j}{|R_{h^t}|}
\]
where all entries of \( \text{adj } R_{h^t} \) and \( p_j \) are integers, \( |R_{h^t}| \) also is an integer, hence \( k = |R_{h^t}| \) will always give an integer \( \lambda \), and almost certainly will be the smallest value to do so; it will do so unless the vector \( \text{adj } R_{h^t} \cdot p_j \) has all its components with a common factor \( > 1 \).

To prove that the solution is unique for the smallest \( k \), assume for some \( k^* \) and \( \lambda^* \) that
\[
kp_j + \sum \lambda^*_i cy_i = k^*p_j + \sum \lambda^*_i cy_i
\]
Then
\[
\sum (\lambda_i - \lambda^*_i) cy_i = (k^* - k) p_j,
\]
and
\[
R_{h^t}^{-1}(\lambda - \lambda^*) = (k^* - k) p_j.
\]
Since \( R_{h^t}^{-1} \) exists
\[
k^* = k \iff \lambda = \lambda^*.
\]
Therefore the solution uniquely determines the smallest \( k \).

If the complement of tuple \( h^t \) contains a pseudo-route \( p^*_i \) which is a combination of \( k \) pseudo-routes where \( k \) is unique, then the product related to \( h^t \) must appear at least \( k^2 \) times in \( \Delta \).
Now we draw a comparison between the above process and that of using pseudo cycles. In the equation

\[ kp_j + \sum_{i} \lambda_i c y_i = PR, \]  

\[ \sum_{i} \lambda_i c y_i \]  

is a pseudo-cycle. If \( k \) is greater than one, say \( a \), then one of the links, \( a \), of the pseudo-cycle must be repeated \( a \) times.

For any tuple \( h_t \), if the complement of \( j_t^a \) contains pseudo-cycle \( \sum_{i} \lambda_i c y_i \), then the product of \( h_t \) will appear at least \( a^2 \) times in \( \Delta \).

Hence, the set of pseudo-cycles can be used in determining if any tuple \( h_t \), produced by the selection process (method 1 section 4.7), appears in \( \Delta \) or not. Simply remove any link \( a \) from \( h_t \) and see if the complement of the remaining tuple, \( j_t^a \), contains a unique pseudo-cycle.

To determine \( \Delta_{w_1}^a \) we need, as in the case where \( M \) is a real regular matrix, to find the set of all pseudo-routes.

Let \( p_i^* \) be a pseudo-route which is a combination of \( k \) routes and includes a link \( a \) which is repeated \( \beta \) times. If the complement of some tuple \( h_t \) contains the pseudo-route \( p_i^* \), and if \( h_t \) is repeated \( a^2 \beta \) times in \( \Delta \), then the multiple of \( h_t \) will appear \( \frac{a^2 \beta}{k} \) times in \( \Delta_{w_1}^a \). In most cases \( k = \alpha \), so the multiple of \( h_t \) will appear \( \alpha \beta \) times in \( \Delta_{w_1}^a \).

We can use the set of pseudo cycles to determine the set of pseudo-routes which are a combination of more than one route, by picking each pseudo-cycle with a repeated link, and combining the pseudo-cycle with a multiple of some route to obtain, a pseudo-route which contains none of the links represented in \( h_t \).
Example

For the graph in Fig (4.7), to obtain a pseudo-route that does not contain links 2 and 5, first we find if there exists a unique pseudo-cycle containing either link 2 or link 5. The set of pseudo-cycles is:

\[ \begin{align*}
\text{cy}_1 &= 1,2,3,-4,-5,-6 \\
\text{cy}_2 &= 2,5,7,-8 \\
\text{cy}_3 &= 1,2,2,3,-4,-6,7,-8 \\
\text{cy}_4 &= 1,3,-4,-5,-5,-6,-7,8
\end{align*} \]

while the set of pseudo-routes (obtained by combining a route with a cycle) is:

\[ \begin{align*}
P_1 &= 1,2,3 \\
P_2 &= 4,5,6 \\
P_3 &= 1,3,-5,-7,8 \\
P_4 &= -2,4,6,-7,8
\end{align*} \]

\( \text{cy}_3 \) is the only pseudo-cycle not containing link 5, likewise \( \text{cy}_4 \) is the only pseudo-cycle not containing link 2.

By choosing \( \text{cy}_3 \) and route \( P_1 \) which contain link 2 and do not contain link 5, a pseudo-route containing neither link 2 nor link 5 can be found from \( P_5 = 2P_1 - \text{cy}_3 = 1,3,4,6,-7,8 \).

Since \( g_2g_5 \) appears 4 times in \( \Delta \), and pseudo-route 1,3,4,6,-7,8 is the combination of 2 routes, then the effect of links 1,3,4,6 and 8 on the cost of travel between origin-destination pair 1, will include the term \( g_2g_5 \) twice (i.e. 4/2 x 1), where 1 reflects the number of times links 1,3,4,6, and 8 are repeated in the pseudo-route. The effect of link 7 will also include \( g_2g_5 \) twice, but with a negative sign.
To check the above

\[ \Delta_{w_1}^1 = \begin{vmatrix} g_1^+ & g_2^+ & g_3 & 0 & g_2 & 0 & -1 & 0 \\
0 & g_4^+ & g_5 & g_6 & g_5 & 0 & 0 & 0 \\
g_2 & g_5 & g_4^+ & g_5 & g_7 & 0 & 0 & -1 \\
0 & 0 & 0 & g_8 & 0 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{vmatrix} \]

= \ g_2 g_4^+ \ + \ 2 g_2 g_5 \ + \ g_2 g_6^+ \ + \ g_4^+ g_5 \ + \ g_4^+ g_7^+ \ + \ g_4^+ g_8^+ \ + \ g_5^+ g_6 \ + \ g_5^+ g_7^+ \ + \ g_5^+ g_8^+ \ + \ g_6^+ g_7^+ \ + \ g_6^+ g_8^+ .

### 4.9 Conclusion

The sign of \( \delta U_{w_1} \) will always be decided by the sign of the numerator \( \Delta_{w_1}^e \) e from eq. (4.10), since the denominator is always positive:

1. Although \( \Delta_{w_1}^e \) may be positive or negative, we know that \( \sum_{eq \in P_{w_1}} \Delta_{w_1}^q = \Delta \), where \( \Delta > 0 \) and \( \sum_{eq \in P_{w_1}} \Delta_{w_1}^q = \Delta_{w_1}^e \).

2. The aim should normally be to make \( \delta U_{w_1} < 0 \) i.e. to reduce any delay on link e, to obtain a negative \( \delta U_{w_1} \).
iii) If $\Delta^e_{w_1} < 0$ (a "Paradoxical" link), then the aim should normally be to make $\varepsilon_e > 0$, or to increase the travel cost on link $e$; this may well be easier in practice, if such a link exists.

Also, the link that will give the largest decrease in total travel cost for small improvements in their travel cost will be those links $e$ with large positive $\Delta^e_{w_1}$ value and the effort required in finding such links is a good investment. Furthermore, improving the most congested link in the network is not necessarily the best policy to follow, since $\Delta^e_{w_1} = \sum_{T \in T} g_a$ where none of the tree complements $T$ contains link $e$. If link $e$ is so congested that $g_e$ overshadows all other $g_a$, then there may well exist a link $b$ with $\Delta^b_{w_1} > \Delta^e_{w_1}$. In that case improving link $b$ may be a better strategy to follow; encouraging users to avoid the problem link rather than altering it.

The above analysis may be used to find the sensitivity of any link $e$ in the network. If $\delta U_{w_1}$ is large then a small error in the congestion function of $e$ will have a significant effect on the calculation of total cost of travel in the network and on the distribution of traffic on the network. Hence it is important to check the accuracy of the congestion function $c_e(f_e)$; conversely if $\delta U_{w_1}$ is small then a small error in $c_e(f_e)$ will have an insignificant effect on the final solution.

In the flow redistribution case the sign of $\delta f_a^e$ will always be decided by the sign of $\Gamma_{ea}$.

i) If $\Gamma_{ea}$ is positive, then any increase in the cost of travel on link $e$ will cause an increase in the flow on link $a$; any decrease in the cost of travel on link $e$ will cause a decrease in the flow on link $a$. 
ii) If $\Gamma_{ea}$ is negative, then any increase in the cost of travel on link e will cause a decrease in the flow on link a; any decrease in the cost of travel on link e will cause an increase in the flow on link a.

iii) $\Gamma_{ea} = \Gamma_{ae}$.

If M is a real regular matrix the selection process, described in section 4.7, can be used to determine $\Delta$. However, for networks with irregular matrix M, the selection process should be used only to find a rough estimate of the effect of a change in the cost of travel on a link on the average cost of travel, and the resulting flow redistribution on other links.

We hope that further research will enable one to determine exactly, or at least close the gap between the upper and lower limits of, the number of times the product of each tuple $h^t$ appear in $\Delta$ ($k^2$ is the lower bound and the number of times the tuple is produced by the selection process squared is the upper bound). At present if $\delta U^w_i$ and $\delta F^a_e$ are required accurately, then they must be found by evaluating the determinant, or by evaluating each individual $|R_{h^t}|$ for $v$-tuples $h^t$. However, pseudo-cycles and pseudo-routes can then be used to determine $\Gamma_{ea}$ and $\Delta^w_i$. Once pseudo-routes and cycles have been found, the effect of a change in the cost of travel of any link on the flow on any other link in the network, and the effect of a change in the cost of travel of any link on the average travel cost become easily and quickly obtainable.
5.1 Introduction

In the previous chapter, the transportation problem with fixed demand was discussed. Now, we shall investigate some aspects of the transportation problem where the demand is elastic. The behaviour of the flow, and the average cost of travel, due to changes in demand will be observed. When certain travel cost constraints are ignored, possible catastrophic behaviour of the point representing user-equilibrium flow will be indicated.

Beckman, McGuire and Winston (1956), made the first computational attempt at solving problems with elastic demand. Wilkie and Stefanek (1971) used both a constrained gradient and a modified Newton-Raphson algorithm to determine the network flows that maximize the equilibrium flows. Gartner (1980b) surveyed earlier work on this topic, and describes three models that can be used to generalize the traffic assignment problem as an equivalent network in which elastic demand functions are represented by appropriate generating links. Hall (1978) showed that for a network, where the cost of travel on a link is dependent on the flow on that link alone, increasing the demand between any origin-destination pair \( i \) causes an increase in the average cost of travel for origin-destination pair \( i \). Fisk (1979) provided an example where increasing the demand between origin-destination pair \( i \) causes a decrease in the average cost of travel for origin-destination pair \( j; i = j \); and a decrease in the total cost of travel in the network. Netter (1972) showed that, in a multimodal network where the cost of travel is not necessarily convex, a multi equilibria solution is possible.

In this section, a linear sensitivity analysis is carried out and the supply and demand equilibrium point is determined. As the demand is increased the flow at the equilibrium point is observed.
5.2 Notation And Assumptions

Consider, for simplicity, a single origin single destination transportation network $N(V,L)$.

The following notation will be used:

- $C_i(F)$ is the cost of travel on route $i$, when the route flow distribution is $F$.
- $c_a(f)$ is the cost of travel on link $a$, when the link flow distribution is $f$; the cost of travel on a link is assumed dependent on the flow distribution in the whole network.
- $u$ is the average cost of travel per user
- $d$ is the elastic demand for travel

Let the demand $d$ be a decreasing function of the average cost of travel $u$; $d = \alpha u$, where $\alpha$ is a measure of the car ownership of the population.

A user-equilibrium flow distribution, according to Wardrop's first principle is assumed. Therefore:

$$C_i(F) = u \leq C_j(F), \text{ for } i \text{ and } j \text{ where } F_i > 0 \text{ and } F_j = 0$$

Here $u = C_i(F) = \sum_{a} \delta_{ai} c_a(f)$

and $d = \sum_{i=1}^{n} F_i$, where $n =$ number of used independent routes joining the origin to the destination in $N(V,L)$

and $\delta_{ai} = \begin{cases} 1 & \text{if link } a \text{ belongs to route } i \\ 0 & \text{otherwise} \end{cases}$
Now, suppose that the car ownership factor $a$ increases slightly to $a + \delta a$, without causing any change in the used routes; this will trigger off the following changes:

\[ u \rightarrow u + \delta u = u^* \text{ say, and} \]

\[ F \rightarrow F + \delta F = F^*. \]

Then \( d \rightarrow d + \delta d = (a + \delta a) f \left( u + \delta u \right) = d^*; \)

therefore \( \delta d = a \delta u f'(u) + \delta a \cdot f(u). \) .... (5.1).

Also \( C_i(F^*) = u + \delta u = \sum \delta a_i \left[ c_a(f + \delta f) \right] \)

\[ = \sum_a \delta a_i \left[ c_a(f) + \sum_b \delta f_b \frac{\partial c_a(f)}{\partial f_b} \right] \]

\[ = \sum_a \delta a_i c_a(f) + \sum_b \delta f_b \sum_j \delta f_j \frac{\partial c_a(f)}{\partial f_b} \delta_{bj} \]

\[ = \delta a_i c_a(f) + \sum_j \delta f_j \theta_{ij}. \]

Hence, \( \delta u = \sum_j \theta_{ij} \delta f_j \) \( \forall i ) \) .... (5.2)

where \( \theta_{ij} = \sum_{a,b} m_{ab} \delta a_i \delta_{bj} \)

and \( m_{ab} = \frac{\partial c_a(f)}{\partial f_b}; \)

also \( \sum_i \delta F_i = \delta d. \) .... (5.3)

Assuming that we have \( n \) used routes, we get \( n + 1 \)
equations 5.2 and 5.3, and by Cramer's rule

\[ \delta u = \frac{\Delta^*}{\Delta} \delta d \]  .... (5.4)

where

\[ \Delta^* = \left| \begin{array}{cccc}
\theta_{11} & \theta_{12} & \cdots & \theta_{1n} \\
\theta_{21} & \theta_{22} & \cdots & \theta_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
\theta_{n1} & \theta_{n2} & \cdots & \theta_{nn}
\end{array} \right| \]
In a multi-origin multi-destination network suppose the demand for origin-destination pair $w_i$ is increased by $\delta d_{w_i}$, while the demand between other origin-destination pairs is unchanged. The analogues to equations 5.1, 5.2, 5.3 and 5.4 are:

$\delta d_{w_i} = \alpha_{w_i} \delta u_{w_i} f'(u_{w_i}) + \delta \alpha_{w_i} f(u_{w_i})$, \hspace{1cm} (5.5)

$\delta u_{w_i} = \sum_j \theta_{ij} \delta F_j$, \hspace{1cm} (5.6)

$\sum_{i \in P_{w_i}} \delta F_i = \delta d_{w_i}$, \hspace{1cm} (5.7)

where $\delta d_{w_i} = 0$ for origin-destination pair(s) $w_j \neq w_i$,

and $\delta u_{w_j} = (-1)^{i+j} \Delta_{ij}^* \delta d_{w_i}$, \hspace{1cm} (5.8)

for $j = i$

where

$\delta u_{w_i} = \frac{\Delta_{ii}^*}{\Delta} \delta d_{w_i}$, \hspace{1cm} (5.9)

with a new link $i$ joining the origin to the destination of origin-destination pair $w_i$. The summation is over all spanning pseudo-minimum link trees.

where

\[
\Delta = \begin{bmatrix}
\theta_{11} & \cdots & \theta_{1n} & -1 \\
\vdots & \ddots & \vdots & \vdots \\
\theta_{n1} & \cdots & \theta_{nn} & -1 \\
1 & \cdots & 1 & 0
\end{bmatrix} - \text{B}^\text{tr}
\]
and

\[ \Delta^*_i = \begin{bmatrix} 0_1 & \cdots & 0_n \end{bmatrix} - B_{i} \]

where \( B_i \) denotes matrix \( B \) with the \( i \)th row removed.

When the cost functions are symmetric; that is, \( \frac{\partial c_a}{\partial f} = \frac{\partial c_b}{\partial f} \)

\[ \Delta = A^T G A - B^T \]

\[ \Delta^*_i = A^T G A - B_{i}^T \]

clearly \( \Delta^*_i = \Delta^*_j \). Here \( G \) is diagonal if the cost of travel on a link is dependent of the volume of traffic on that link alone.

**Theorem 5.1**

For a diagonal matrix \( G \), and graph \( N(V,L) \)

\[ \Delta^*_i = \sum_{T} a_T G_a \]

where \( T \) is the complement of the spanning tree \( T \) of the undirected graph \( N^* \), where graph \( N^* \) equals graph \( N \) with a new link \( a \) joining the origin to the destination of origin-destination pair \( w_i \). The summation is over all spanning pseudo-trees \( T \) that contain link \( a \).

**Proof.**

Let \( A \) be the link-route incidence matrix for graph \( N \), \( G \) be its diagonal matrix, and \( B \) is the \( Z \times Q \) matrix of \( N \) whose \((i,p)\) entry equals 1 if route \( p \) joins 0-D pair \( w_i \) and
0 otherwise. Arrange the order of the 0-D pairs so that 0-D pair \( w_1 \) is numbered 0-D pair one.

The determinant of the cyclic matrix for graph \( N_1 \) is

\[
\Delta = \begin{vmatrix}
g_L & 0 & 0 & 0 & \cdots & 0 \\
0 & A_{\tr G} & A & 0 & \cdots & 0 \\
0 & 0 & A_{\tr G} & A & 0 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots \\
-\bar{B} & \cdots & \cdots & \cdots & 0 & \cdots \\
0 & \cdots & \cdots & \cdots & \cdots & \cdots \\
\end{vmatrix}
\]

where \( \bar{B} \) is a \((Q \times Q + 1) \) matrix where:

- column 1 of \( \bar{B} \) = \( \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \) and
- column 2 to column \( Q+1 \) of \( \bar{B} \) = column 1 to column \( Q \) of \( B \).

Now \( \Delta = \sum_{T} \prod_{a \in T} g_a \), where the summation is over all spanning pseudo-trees of \( N_1^* \), (theorem 4.5.1)

Hence

\[
\Delta = \sum_{T_1} \prod_{a \in T_1} g_a + \sum_{T_2} \prod_{a \in T_2} g_a
\]

where \( T_1 \) are spanning pseudo-trees that contain link \( l \), and \( T_2 \) are spanning pseudo-trees that do not contain link \( l \).

From the expansion of determinant \( \Delta \) (equation 5.12), by the top row then the first column:

\[
\Delta = g_L \begin{vmatrix}
A_{\tr G} & -B_{\tr} \\
\bar{B} & 0 \\
\end{vmatrix}
+ \begin{vmatrix}
A_{\tr G} & \bar{B}_{\tr} \\
\bar{B} & 0 \\
\end{vmatrix}
\]

Hence \( t_1 \) will appear in both.
Therefore

\[
\begin{pmatrix}
A^{\text{tr}}GA & -B_1^{\text{tr}} \\
B_1 & 0
\end{pmatrix}
= \sum_{T_1 \in T_1} \pi_{T_1} g_a
\]

since it does not contain the parameter \( g_a \).

If \( G \) is diagonal then \( \Delta \) and \( \Delta^{*}_{ij} \) are both positive, hence increasing (decreasing) \( \delta d \) will cause \( \delta u \) to increase (decrease) as in Hall (1978).

\( \Delta^{*}_{ii} \) and \( \Delta \) are positive providing that \( A^{\text{tr}}GA \) is symmetric positive definite (Dafermos & Nagurney 1984).

For graph \( N \), let \( N_1 \) be graph \( N \) with O-D pair \( w_i \) joined by link \( \ell_i \). Let \( M_1 \) be the cycle-link incidence matrix for graph \( N_1 \). By suitable row and column subtraction operations, we obtain \( \Delta^{*}_{ii} = M_1^{\text{tr}} GM_i^* \), where matrix \( M_1^* \) is equal to matrix \( M_1 \) with the column representing link \( \ell_i \) removed.

**Theorem 5.2**

For a graph \( N \), with a diagonal matrix \( G \), any tuple product \( t \) appearing in \( \Delta^{*}_{i} \) must also appear in \( \Delta^{*}_{ii} \) and \( \Delta^{*}_{jj} \).

**Proof**

By suitable row subtraction operations and corresponding column operations, we obtain

\[
\Delta^{*}_{ij} = |M_1^{\text{tr}} GM_j^*|
\]

Let \( t_1 \) be a \( v+1 \)-tuple product appearing in \( \Delta^{*}_{i} \), then

\[
|R_{t_1}^i| \neq 0 \text{ and } |R_{t_1}^j| \neq 0,
\]

where \( R_{t_1}^i \) is a major square submatrix of \( M_1^* \).

so

\[
|R_{t_1}^i|^2 \neq 0 \text{ and } |R_{t_1}^j|^2 \neq 0
\]

hence \( t_1 \) will appear in both
\[ \Delta_{ij}^* = |M_i^*trGM_i^*| \quad \text{and} \quad \Delta_{jj}^* = |M_j^*trGM_j^*| \]

Likewise if \( t_2 \) does not appear in \( \Delta_{ij}^* \) then either
\[ |R_i^{t_2}| = 0 \quad \text{or} \quad |R_j^{t_2}| = 0; \quad \text{hence} \]
\[ |R_i^{t_2}|^2 = 0 \quad \text{or} \quad |R_j^{t_2}|^2 = 0. \]
So that \( t_2 \) does not appear in either \( \Delta_{ii}^* \) or in \( \Delta_{jj}^* \).

As a result any tuple product appearing in \( \Delta_{ij}^* \) must also appear in \( \Delta_{ii}^* \) and \( \Delta_{jj}^* \), any tuple product appearing in \( \Delta_{ii}^* \) and \( \Delta_{jj}^* \) will appear in \( \Delta_{ij}^* \).

For a "Real Regular" matrix \( M_{ij} \),
\[ \Delta_{ij}^* = \Delta_{ii}^* \cap \Delta_{jj}^* \]

If \( M_{ij} \) is not regular then for any tuple product \( t \) appearing in \( \Delta_{ij}^* \), the number of times \( t \) is repeated in \( \Delta_{ij}^* \) is given by
\[ \sqrt{\text{the number of times } t \text{ is repeated in } \Delta_{ii}^*} \times \sqrt{\text{the number of times } t \text{ is repeated in } \Delta_{jj}^*} \]

5.3 Effect Of A Change In Demand On The Link Flow

Assume the demand between O-D pair \( w_j \) is increased, and the cost functions are symmetric. From equations 5.6 and 5.7 we get
\[ \delta F_i = \frac{\Delta_i^*}{\Delta} \delta w_j \]
where
\[
\Delta_i^* = \begin{bmatrix} \text{A}^{tr}CA & -B^{tr} \\ \text{B} & 0 \end{bmatrix}
\]

with column \( i \) replaced by a \((n + Z) \times 1\) vector, whose \( n + W_j \) entry equals one and all other entries are zero, here \( n \) is the number of used routes and \( Z \) is the number of origin destination pairs.
Therefore, 
\[ \Delta^*_i = \begin{vmatrix} A^{tr}G_A & -B^{tr} \\ B_i & 0 \end{vmatrix} \]

where \( \tilde{A} = A \) with entries of column \( i \) equal to zero
\( \tilde{B}_i = B \) with the entries of column \( i \) equal to zero except that entry \((w_j,i)\) equals 1.

Since \( f_a = \sum_i \delta a_i \tilde{F}_i \) then,
\[ \delta f_a = \sum_i \delta a_i \frac{\Delta^*_i}{\Delta} \delta w_j \]
so
\[ \delta f_a = \begin{vmatrix} A^{tr}G_A & -B^{tr} \\ B_i & 0 \end{vmatrix} \frac{\Delta^*_i}{\Delta} \delta w_j \]

where \( B_i \) is equal to \( B \) with row \( i \) replaced by \( \xi^{tr} \), where \( \xi \) is a \( Q \times 1 \) vector, the transpose of row \( a \) of matrix \( A \).

Taking the transpose yields,
\[ \delta f_a = \begin{vmatrix} A^{tr}G_A & -B^{tr} \\ B & 0 \end{vmatrix} \frac{\Delta^*_i}{\Delta} \delta w_j \] ... (5.13)

The numerator determinant is exactly that in equation (4.10).

From Chapter 4 we know that if
\[ \begin{vmatrix} A^{tr}G_A & -B^{tr} \\ B & 0 \end{vmatrix} \]
is negative then link \( a \) will be paradoxical; if
\[ \begin{vmatrix} A^{tr}G_A & -B^{tr} \\ B & 0 \end{vmatrix} \]
is positive link \( a \) is not paradoxical. Therefore if link \( a \) is not paradoxical, then from equation (5.13), \( \delta f_a \) will be positive; if link \( a \) is paradoxical, \( \delta f_a \) will be negative. So a paradoxical link is paradoxical in two
ways - if the cost of travel on the link is increased the overall travel cost is decreased, and if the flow through the network is increased, the flow through the link is decreased.

5.4 Catastrophic Behaviour Of The Equilibrium Point

Providing that $\delta \alpha$ is positive, $\delta \alpha f(u)$ will also be positive, and since the demand is decreasing; $f'(u)$ is negative. From equations (5.1) and (5.4), we deduce the following:

$$\frac{\Delta}{\Delta^*} \delta u = \alpha \delta uf'(u) + \delta \alpha f(u),$$

$$\delta u(\frac{\Delta}{\Delta^*} - \alpha f'(u)) = \delta \alpha f(u) > 0.$$

Then either

A) $\delta u > 0$ and $\frac{\Delta}{\Delta^*} - \alpha f'(u) > 0$, or,

B) $\delta u < 0$ and $\frac{\Delta}{\Delta^*} - \alpha f'(u) < 0$.

Case (A) is true when

i) $\frac{\Delta}{\Delta^*}$ is positive; the cost function is increasing, (Fig 5.1a), or when

ii) $\frac{\Delta}{\Delta^*}$ is negative, and $\left|\frac{\Delta}{\Delta^*}\right| < \left|\alpha f'(u)\right|$: the cost function is decreasing at a faster rate than the demand function, (Fig 5.1b).

Case (B) is true when $\frac{\Delta}{\Delta^*}$ is negative, and

$$\left|\frac{\Delta}{\Delta^*}\right| > \left|\alpha f'(u)\right|;$$

the cost function is decreasing at a slower rate than the demand function, (Fig 5.1c).
If the link cost function is dependent on the flow on that link alone, then $\Delta$ will be positive, and $\Delta$ will also be positive (theorem 4.5.1). Hence, if the demand increases, the average cost of travel will increase too, as proved by Hall (1978). Case A(ii), where the cost function is decreasing at a faster rate than the demand function, can occur only when $|f'(u)|$ is small. This might happen in practice when two modes of transport are used between the origin and the destination, and the cost of travel on one mode is almost independent of the cost of travel on the other mode; for example, where a road network is paralleled by a high capacity rail link. Mogridge (1985) demonstrates the occurrence of Thomson's contention: the quality of car travel is at equilibrium with its competitors. In the case of London, the alternative mode is the rail system. Mogridge concludes that any increase in road capacity in London will not affect speed; only changes in the rail service could improve road speeds.

As in Beckman et al. (1956), the equilibrium solution is called stable if: when a small deviation from equilibrium occurs, then the equilibrium forces will counteract such deviation and equilibrium is restored. Note that the solution of: Case A(ii) (fig 5.1b), is unstable – a small increase in the flow will cause the cost of travel to become lower than the expected equilibrium cost, so that
the demand for travel will increase, causing a further increase in flow and so on. Likewise a small decrease in flow will cause the cost of travel to become higher than the expected equilibrium cost, so that the demand for travel decreases, causing a further decrease in flow ... .

In figure 5.2, the equilibrium point is \((d_1, u_1)\). If the flow increases to \(d_2\), the cost of travel will be \(u_2\). However, at the demand \(d_2\), users are prepared to pay the cost \(u_2 > u_3\), hence more users will be tempted, and the flow increases to \(d_3\) and so on.

Consider the single origin-destination network of flow and demand. The diagram illustrated in Figure 5.2 shows the relationship between flow and demand, where the cost of travel is represented by the function \(c(u)\). The equilibrium point \((d^*, u^*)\) is such that at this point, the demand equals the supply, and the cost of travel is equal to the expected equilibrium cost.

Case B (fig 5.1c), is stable - a small increase (decrease) in the flow will cause the cost of travel to become higher (lower) than the expected equilibrium cost, hence the demand for travel will decrease (increase) and equilibrium is restored eventually. Case A(i) (fig 5.1a) could be either stable or unstable; depending on the slope of the demand and cost functions.

Suppose we have a network in which \(-\frac{\Delta}{\Delta*}\) is negative for some range of values of demand and positive elsewhere, and a demand function \(d = \alpha f(u)\), where \(|\alpha f'(u)| > \frac{\Delta}{\Delta*}\) when \(-\frac{\Delta}{\Delta*}\) is negative.

Then as \(\alpha\) increases the equilibrium point will trace the path EB (in Figure 5.3) then jump to C and move along CF; likewise as \(\alpha\) decreases the equilibrium point will trace the path FD then jump to E and trace the path EA.
Where these jumps occur, the cost of travel to each user changes slightly. However, the demand for travel will increase or decrease drastically, causing a major variation in flow distribution in the network.

The following example illustrates how this behaviour can occur if the cost on a link is reduced by investment.

**Numerical Example**

Consider the single origin-destination network of four nodes and five links shown in figure 5.4. Assume the following link congestion functions:

\[
\begin{align*}
    c_{OA}(f) &= 5f_1 \\
    c_{OB}(f) &= 6f_2 + 5f_5 + 50 \\
    c_{AD}(f) &= 6f_3 + 5f_5 + 50 \\
    c_{BD}(f) &= 5f_4 \\
    c_{AB}(f) &= 5f_2 + 5f_3 + \frac{11}{1 + 20n} \cdot f_5 + 10 - 5n.
\end{align*}
\]

Here the parameter \( n \) represents some investment on improving link AB.
The three routes from 0 to D are OAD, OBD and OABD, and the cost associated with each route is:

\[ C_{OAD} = C_1 = 11F_1 + 10F_3 + 50 \]
\[ C_{OBD} = C_2 = 11F_2 + 10F_3 + 50 \]
\[ C_{OABD} = C_3 = 10F_1 + 10F_2 + (10 + \frac{11}{1+2.3n})F_3 + 10-5n \]

where \( F_1, F_2 \) and \( F_3 \) are the flows on the three routes.

If the total traffic demand is \( d \), so that \( f_1 + f_2 + f_3 = d \), the user-equilibrium assignment of flows on the network depends on the parameters \( n \) and \( d \). We shall plot the cost of travel, \( u \), against demand \( d \) for different values of \( n \).

Writing \( \frac{11}{1+2.3n} \) as \( \theta \), and assuming that \( \theta > 4.5 \) (or \( n < 0.628 \)), the user-optimal assignment is:

**Case I**

\[ F_1 = F_2 = \frac{d}{2}, \quad F_3 = 0; \]  
routes OAD and OBD are used. This holds for \( d \geq \frac{(80 + 10n)}{9} \), and \( u = 50 + \frac{11}{2d} \).

**Case II**

\[ F_1 = F_2 = \frac{2 \theta d - 10n - 80}{2(28-9)}, \quad F_3 = \frac{80 + 10n - 9d}{28-9}; \]

all three routes are used. This holds for

\[ \frac{40 + 5n}{\theta} \leq d \leq \frac{80 + 10n}{9}, \quad u = \frac{100 \theta - 90 + 45n + (11 \theta - 90)d}{28-9} \]
Case III

$F_1 = F_2 = 0$ and $F_3 = d$; only route $OABD$ is used. This holds for $d \leq \frac{40 + 5n}{10}$, and $u = 10 - 5n + (10 + \theta)d$

It appears that $u$ is a continuous function of $d$ for each value of $n$, and in cases I and III, is always an increasing function of $d$. But in Case II, $u$ increases with $d$ if $n < \frac{31}{207} = 0.149$, but it decreases as $d$ increases if $n > \frac{31}{207}$. See Fig (5.5).
Let the demand function be \( d = \alpha (1660.2 - 16.5306u) \), the equilibrium point for the cost function, with \( n = 0.155 \) will be as follows:

<table>
<thead>
<tr>
<th>( \alpha )</th>
<th>( d )</th>
<th>( u )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>5.0197</td>
<td>100.12826</td>
</tr>
<tr>
<td>1.4</td>
<td>5.024</td>
<td>100.21467</td>
</tr>
<tr>
<td>2.0</td>
<td>5.028</td>
<td>100.27984</td>
</tr>
<tr>
<td>2.2</td>
<td>9.1238</td>
<td>100.18104</td>
</tr>
<tr>
<td>2.6</td>
<td>9.13</td>
<td>100.215</td>
</tr>
</tbody>
</table>

\( F_3 = d, F_1 = F_2 = 0 \)

\( F_1 = F_2 \) & \( F_3 = 0 \)

By reversing the process, starting at \( \alpha = 2.6 \):

<table>
<thead>
<tr>
<th>( u )</th>
<th>( \alpha )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.6</td>
<td>9.13</td>
</tr>
<tr>
<td>2.4</td>
<td>9.127</td>
</tr>
<tr>
<td>2.0</td>
<td>9.11929</td>
</tr>
<tr>
<td>1.4</td>
<td>9.09</td>
</tr>
<tr>
<td>1.0</td>
<td>9.0696</td>
</tr>
<tr>
<td>0.9</td>
<td>5.0179</td>
</tr>
</tbody>
</table>

Therefore as \( \alpha \) increases from 1 to 2.0 the demand and average cost of travel will increase gradually from 5.0197 to 5.028 and 100.12826 to 100.2798 respectively. A further increase in \( \alpha \) to 2.2 will cause the average cost of travel to decrease to 100.1804, however the demand for travel will increase drastically to 9.1238. However as \( \alpha \) starts to decrease from 2.6 to 2.0 and then to 1, the
demand will not change back immediately to 5.028; the demand for travel will decrease from 9.13 to 9.11929 and the average cost of travel from 100.215 to 100.156. Once $\alpha$ decreases to 0.9 the demand decreases drastically to 5.0179 and the cost of travel increases to 100.094. Concluding that as $\alpha$ changes from 0.9 to 2.6 and back to 0.9 the equilibrium point follows a standard fold catastrophe.

5.5 CONCLUSION.

The effect of a change in the demand between origin destination pair $w_i$, $d_{w_i}$, on the cost of travel between origin destination pair $w_j$ is determined by $\Delta_i$, $\Delta^*_{ij}$ and $\delta d_{w_j}$. We have established that $\Delta^*_{ij} = \Delta^*_{ji}$, and established a relationship between $\Delta^*_{ii}$, $\Delta^*_{jj}$ and $\Delta^*_{ij}$.

Then we showed that as the flow in the network increases, the flow on a paradoxical link decreases.

Finally, given certain cost and demand function conditions, the point representing the user equilibrium flow will jump suddenly, causing a drastic change in demand and flow distribution. However the cost of travel to each user is continuous. In a complex network with several competing modes of transport, this phenomenon may be widespread. Taken with Thomson's contention, it shows the importance of not dealing with any mode in isolation.
APPENDIX

Programme: Dijkstra 1

This programme finds the shortest route between an origin and a destination node(s) by using Dijkstra's algorithm. It is designed for networks with a single mode of transport and a fixed cost of travel on the links.

Programme: Dijkstra 2

This programme finds the shortest route between an origin and a destination node(s), by using Dijkstra's algorithm. It is designed for networks with a multi-mode transportation and a fixed cost of travel on the links.

Programme: Floyd 1

This programme finds the shortest routes between all the nodes in the network by using Floyd's algorithm. It is designed for networks with a single mode of transport and a fixed cost of travel on the links.

Programme: Floyd 2

This programme finds the shortest routes between all the nodes in the network by using Floyd's algorithm. It is designed for networks with a multi-mode transportation and a fixed cost of travel on the links.

Programme Dijkstra 2 and Floyd 2, were used for obtaining the shortest routes of travel between all the nodes (stations) in the London Underground network. Floyd 2 was faster in obtaining the shortest routes.
Programme: Dijkstra 1

This programme finds the shortest route between an origin and a destination node, by using Dijkstra's algorithm. It is designed for networks with a single mode of transport and a fixed cost of travel on the links.

Programme: Dijkstra 2

This programme finds the shortest route between an origin and a destination node, by using Dijkstra's algorithm. It is designed for networks with a multi-mode transportation, and a fixed cost of travel on the links.

Programme: Floyd 1

This programme finds the shortest routes between all the nodes in the network by using Floyd's algorithm. It is designed for networks with a single mode of transport and a fixed cost of travel on the links.

Programme: Floyd 2

This programme finds the shortest routes between all the nodes in the network by using Floyd's algorithm. It is designed for networks with a multi-mode transportation, and a fixed cost of travel on the links.

Programmes Dijkstra 2 and Floyd 2, were used for obtaining the shortest routes of travel between all the nodes (stations) in the London Underground network. Floyd 2 was faster in obtaining the shortest routes.
DIJKSTRA 1

INTEGER IW(50,50), IRR(50,50,50), IG(50), IW1(50,50)
*, IW2(50,50), IRR1(50,50,50), IP1(50), C(50,50,10)
WRITE(6,300)

300 FORMAT(IX, 'PRINT NUMBER OF NODES IN I2 FORMAT')
READ(5,100)IN
DO 303 I=1,IN
DO 304 J=1,IN
IF(I.EQ.J)GOTO 304
IW(I,J)=999
304 CONTINUE
303 CONTINUE
DO 404 II=1,130
READ(15,110)(I,J,IW(I,J),(C(I,J,IKK),IKK=1,9))
404 CONTINUE
110 FORMAT(12I2)

DO 101 IR=1,IN
DO 46 I=1,IN
DO 55 J=1,IN
IW1(I,J)=IW(I,J)
55 CONTINUE
46 CONTINUE
100 FORMAT(I2)

DO 77 IE=1,IN
IF(CIE.EQ.DGOTO 70
DO 10 I=1,IN
DO 51 II=1,IN
DO 52 JJ=1,IN
IF(C(IR,IS,II).BQ.O)GOTO 51
IF(C(IS,I,JJ).BQ.O)GOTO 52
IF(C(IR,IS,II).BQ.C(IS,I,JJ))GOTO 95
52 CONTINUE
51 CONTINUE
IF(ISH.EQ.1)GOTO 96
IW2(IR,I)=IW1(IR,I)
GOTO 10
95 IF(IW1(IR,I).GT.(IW1(IR,IS)+IW1(IS,I)))GOTO 90
IW2(IR,I)=IW1(IR,I)
10 CONTINUE
GOTO 70
90 IP1(I)=IS
L=L+1
ISH=1
C(IR,I,L)=C(IS,I,JJ)
GOTO 52
96 IW1(IR,I)=(IW1(IR,IS)+IW1(IS,I))
IW2(IR,I)=IW1(IR,I)
L=0
ISH=0
GOTO 10
70 DO 40 I=1,IN
IF(I.EQ.IR)GOTO 40
IF(IW1(IR,I).LT.IT)GOTO 40
DO 33 IZ=1,IE
IF(I.EQ.IG(IZ))GOTO 40
33 CONTINUE
IF(IW1(IR,I).LE.M)GOTO 11
CONTINUE
IG(IE)=IS
IT=M
M=1000
GOTO 89

M=IW1(IR,I)
IS=I
GOTO 40

IX=IS
DO 67 I=1,IN
IPP=IN-I
IRR1(IR,IS,IPP)=IX
IX=IP1(IX)
IF(IX.EQ.0)GOTO 24
67 CONTINUE
24 IPP=IPP-1
IRR1(IR,IS,IPP)=IR

CONTINUE
DO 17 IM=1,IN
IP1(IM)=0
IG(IM)=0
17 CONTINUE
IPP=0
IT=0
IS=0
M=1000

CONTINUE
DO 103 IR=1,IN
DO 7 8 IE=1,IN-1
IF(CIE.EQ.0)GOTO 71
DO 20 1=1,IN
IF(IW2(IR,I).GT.(IW2(IR,IS)+IW2(IS,I)+5))GOTO 91
20 CONTINUE
GOTO 71
91 IP1(I)=IS
IW2(IR,I)=IW2(IR,IS)+IW2(IS,I)+5
GOTO 20

71 DO 44 I=1,IN
IF(I.EQ.IR)GOTO 44
IF(IW2(IR,I).LT.IT)GOTO 44
DO 34 IZ=1,IE
IF(I.EQ.IG(I2))GOTO 44
34 CONTINUE
IF(IW2(IR,I).LE.M)GOTO 12
44 CONTINUE
IG(IE)=IS
IT=M
M=1000
GOTO 88

M=IW2(IR,I)
IS=I
GOTO 44
78 CONTINUE
GOTO 104
88 WRITE(16,150)IR,IS,IW2(IR,IS)
150 FORMAT(IX, 'MIN COST OF TRAVEL FROM ORIGIN', * ,I3,IX, 'TO DESTINATION',I3,IX, IS ',I6)
IX=IS
DO 66 I=1,IN
IPP=IN-I
IRR(IR, IS, IPP)=IX
IX=IP1(IX)
IF(IX.EQ.0)GOTO 23

CONTINUE

23 IPP=IPP-1
IRR(IR, IS, IPP)=IR
DO 999 J=1,IN
IF(IRR(IR, IS, J).EQ.0)GOTO 999
IK=J+1
WRITE(16,1000)((C(IRR(IR, IS, J), IRR(IR, IS, IK), MM), MM=1,9)
*, IRR(IR, IS, J), IRR(IR, IS, IK), IRR1(IRR(IR, IS, J), IRR(IR, IS, IK), IN-2)
IF(IRR(IR, IS, IK).EQ.IS)GOTO 78

1000 FORMAT(IX, 'USING LINES', IX, 'FROM IX TO IX VIA IX')

999 CONTINUE

104 DO 27 IM=1,IN
IP1(IM)=0
IG(IM)=0
CONTINUE

27 IPP=0
IT=0
IS=0
M=1000

CONTINUE

103 STOP
END
DIJKSTRA 2

INTEGER IW(50,50), IRR(50,50,50), IG(50), IW1(50,50)
  *, IW2(50,50), IRR1(50,50,50), IPI(50), C(50,50,10)
  *, IW3(50,50), CI(50,50,10), C2(50,50,10)
WRITE(6,300)

300 FORMAT(IX, 'PRINT NUMBER OF NODES IN I2 FORMAT')
READ(5,100)IN
DO 303 I=1,IN
  DO 304 J=1,IN
    IF(I.EQ.J)GO TO 304
    IW(I,J)=999
    IW2(I,J)=999
  CONTINUE
303 CONTINUE
M=5000
DO 404 II=1,M
  READ(15,110)(I,J,IW(I,J),(C(I,J,IKK).IKK=1,9))
404 CONTINUE
110 FORMAT(12I2)
DO 888 IPIP=1,9
  DO 101 IR=1,IN
    DO 46 I=1,IN
      DO 55 J=1,IN
        IW1(I,J)=IW(I,J)
      CONTINUE
      DO 54 K=1,9
        C1(I,J,K)=C(I,J,K)
      CONTINUE
  CONTINUE
54 CONTINUE
55 CONTINUE
46 CONTINUE
100 FORMAT(12)
DO 77 IE=1,IN
  IF(IE.EQ.1)GO TO 60
    DO 10 I=1,IN
      DO 51 II=1,9
        DO 52 JJ=1,9
          IF(IR.BQ.I)GO TO 10
          IF(C1(IR,IS,II).NE.IPIP)GO TO 51
          IF(C1(IS,I,JJ).NE.IPIP)GO TO 52
            DO 444 KIK=1,9
              IF(C1(IR,I,KIK).EQ.IPIP)GO TO 95
            CONTINUE
            GOTO 90
          52 CONTINUE
        51 CONTINUE
      50 CONTINUE
    70 CONTINUE
  95 IF(IW1(IR,I).GT.(IW1(IR,IS)+IW1(IS,1)))GO TO 90
    DO 554 IJJ=1,IE
      IF(I.EQ.IG(IJJ))GO TO 10
    CONTINUE
  GOTO 90
52 CONTINUE
GOTO 10
51 CONTINUE
GOTO 70
95 IF(IW1(IR,I).GT.(IW1(IR,IS)+IW1(IS,1)))GO TO 90
  DO 554 IJJ=1,IE
    IF(I.EQ.IG(IJJ))GO TO 10
554 CONTINUE
C1(IR,I,1)=IPIP
GOTO 10
90 IPI(1)=IS
  C1(IR,I,1)=IPIP
  IW1(IR,I)=IW1(IR,IS)+IW1(IS,1)
10 CONTINUE
GOTO 70
60 DO 400 IJ=1,IN
   IF(IJ.EQ.IR)GOTO 400
   DO 440 MM=1,9
      IF(C1(IR,IJ,MM).EQ.IPIP)GOTO 450
500 CONTINUE
   GOTO 400
450 IF(IW1(IR,IJ).LE.M)GOTO 14
500 CONTINUE
   IF(ISH.EQ.1)GOTO 767
   GOTO 101
14 M=IW1(IR,IJ)
   IS=IJ
   ISH=1
   GOTO 400
70 DO 40 I=1,IN
   IF(I.EQ.IR)GOTO 40
   DO 441 MM=1,9
      IF(C1(IR,I,MM).EQ.IPIP)GOTO 616
441 CONTINUE
   GOTO 40
616 IF(IW1(IR,I).LT.IT)GOTO 40
      DO 33 IZ=1,IE
         IF(I.EQ.IG(IZ))GOTO 40
33 CONTINUE
   IF(IW1(IR,I).LE.M)GOTO 11
40 CONTINUE
767 IG(IE)=IS
   IF(IG(IE).EQ.IG(IE-1))GOTO 666
   IT=M
   M=5000
   ISH=0
   GOTO 89
11 M=IW1(IR,I)
   IS=I
   GOTO 40
89 IF(IW1(IR,IS).LT.IW2(IR,IS))GOTO 727
   IF(IW1(IR,IS).EQ.IW2(IR,IS))GOTO 722
    77 CONTINUE
27 IW2(IR,IS)=IW1(IR,IS)
   DO 599 IM=1,9
      C2(IR,IS,IM)=0
599 CONTINUE
   C2(IR,IS,1)=IPIP
777 IX=IS
   DO 67 I=1,IN
      IPP=IN-I
      IRR1(IR,IS,IPP)=IX
      IX=IP1(IX)
      IF(IX.EQ.0)GOTO 24
67 CONTINUE
24 IPP=IPP-1
   IRR1(IR,IS,IPP)=IR
77 CONTINUE
   GOTO 666
722 DO 771 MM=1,9
   IF(C2(IR,IS,MM).EQ.0)GOTO 775
771 CONTINUE
C2(IR, IS, MN) = IPP
GOTO 777

DO 17 IM = 1, IN
   IP1(IM) = 0
   IG(IM) = 0
CONTINUE

M = 5000
CONTINUE
DO 103 IR = 1, IN
   DO 331 I = 1, IN
      DO 332 J = 1, IN
         IW3(I, J) = IW2(I, J)
      CONTINUE
   CONTINUE

M = 5000
CONTINUE
DO 106 IIS = 1, IN
   WRITE(16, 150) IR, IIS, IW3(IR, IIS)
do 999 J = 1, IN
   FORMAT(IX, ' MIN COST OF TRAVEL FROM ORIGIN')
* 13, IX, 'TO DESTINATION', 13, IX, 'IS' J6
IF (IRR(IR, IIS, J) .EQ. 0) GOTO 999
IK = J + 1
WRITE (16, 1000) (C2 (IRR(IR, IIS, J), IRR(IR, IIS, IK), MM), MM = 1, 9)
* 1RR(IR, IIS, J), IRR(IR, IIS, IK), IRR(I, IIS, J), IRR(IR, IIS, IK)
* , IN-2)
IF (IRR(IR, IIS, IK) .EQ. IIS) GOTO 106
1000 FORMAT (IX, ' USING LINES', IX, ' FROM', IX, ' TO', IX, ' VIA', IX)
999 CONTINUE
106 CONTINUE
104 DO 27 IM = 1, IN
1PIM = 0
IGM = 0
27 CONTINUE
1PP = 0
IT = 0
IS = 0
M = 5000
103 CONTINUE
STOP
END

10  W11 = W11 + W11
24 CONTINUE
44 CONTINUE
DO 77 X = 1, N
DO 76 J = 1, N
75 CONTINUE
DO 99 H = 1, N
88 CONTINUE
99 CONTINUE
111 XI, J, J = 1
88 CONTINUE
77 CONTINUE
DO 91 X = 1, N
DO 92 X = 1, N
NWRITE (16, 1000) (X, J, W11, J)
1000 FORMAT (IX, 'EACH COST OF TRAVEL FROM ORIGIN '
X, IX, ' TO DESTINATION', IX, IX, IX, IX)
NWRITE (16, 2000) (X, J, H), H = 1, M
2000 FORMAT (IX, 'PAIR OF TRAVEL', IX, IX)
92 CONTINUE
91 CONTINUE
88 CONTINUE
111 STOP
END
FLOYD 1

INTEGER S,X,Q,W(10,10),P(10,10,10)
WRITE(6,300)
300 FORMAT(1X,'PRINT NO OF NODES IN FORMAT')
READ(5,100)N
100 FORMAT(I2)
READ(15,110)((W(I,J),J=1,N),I=1,N)
110 FORMAT(100I2)
DO 21 I=1,N
   DO 22 J=1,N
      IF(W(I,J).LT.99)GOTO 23
   GOTO 22
23 P(I,J)=J
22 CONTINUE
21 CONTINUE
DO 44 Q=1,N
   DO 55 1=1,N
      DO 66 J=1,N
         IF(W(I,J) .GT.(W(I,Q)+W(Q,J)))G0T0 20
      GOTO 66
20 W(I,J)=W(I,Q)+W(Q,J)
   P(I,J)=P(I,Q)
66 CONTINUE
55 CONTINUE
44 CONTINUE
DO 77 I=1,N
   DO 88 J=1,N
      X=I
      DO 99 M=2,N
         R(I,J,M)=P(X,J)
      IF(P(X,J).EQ.J)GOTO 111
      X=P(X,J)
99 CONTINUE
111 R(I,J,1)=I
88 CONTINUE
77 CONTINUE
DO 91 I=1,N
   DO 92 J=1,N
      WRITE(16,1000)I,J,W(I,J)
   WRITE(16,2000)(R(I,J,M),*=1,N)
92 CONTINUE
91 CONTINUE
STOP
END
FLOYD 2

INTEGER S, X, Q, W(50, 50), P(50, 50), R(50, 50, 50), C(50, 50, 10),
* C1(50, 50, 10), C2(50, 50, 10), WI(50, 50), W2(50, 50), P1(50, 50), F,
* F1(50, 50), R1(50, 50, 50)
WRITE(6, 300)
300 FORMAT(IX, ' PRINT NO OF NODES IN FORMAT')
READ(5, 100) N
100 FORMAT(I2)
DO 303 I = 1, N
DO 304 J = 1, N
IF (I .EQ. J) GOTO 304
W(I, J) = 999
W2(I, J) = 999
304 CONTINUE
303 CONTINUE
DO 404 I1 = 1, 130
READ(25, 110) (I, J, W(I, J), (C(I, J, KK), KK = 1, 9))
404 CONTINUE
110 FORMAT(1212)
DO 21 Q = 1, N
DO 22 J = 1, N
IF (Q .EQ. J) GOTO 22
IF (C1(I, Q, II) .NE. IP1P) GOTO 101
775 IF (C1(Q, J, JJ) .NE. IP1P) GOTO 102
DO 73 IP1 = 1, 9
IF (C1(I, J, IP1) .EQ. IP1P) GOTO 7 5
73 CONTINUE
GOTO 7 5
74 IF (W1(I, J) .GE. (W1(I, Q) + W1(Q, J))) GOTO 7 5
GOTO 66
102 CONTINUE
GOTO 66
101 CONTINUE
GOTO 55
75 W1(I, J) = W1(I, Q) + W1(Q, J)
C1(I, J, 1) = IP1P
P1(I, J) = P1(I, Q)
P(I, J) = J
IT(I, J) = 1
DO 881 I=1,N
  DO 882 J=1,N
  IF(IT(I,J).NE.1)GOTO 882
  IF(W1(I,J).LT.W2(I,J))GOTO 903
  IF(W1(I,J).EQ.W2(I,J))GOTO 904
882 CONTINUE
903 W2(I,J)=W1(I,J)
  DO 808 L=1,9
    C2(I,J,L)=0
  808 CONTINUE
C2(I,J,1)=I+I
  GOTO 178
904 DO 905 MN=1,9
  IF(C2(I,J,MN).GT.0)GOTO 906
  905 CONTINUE
  906 C2(I,J,MN)=I+I+I
  GOTO 882
178 X=I
  DO 65 M=2,N
    R1(I,J,M)=P1(X,J)
    IF(P1(X,J).EQ.J)GOTO 112
    X=P1(X,J)
  65 CONTINUE
  112 R1(I,J,1)=I
  GOTO 882
888 CONTINUE
  DO 45 Q=1,N
  DO 56 I=1,N
  DO 67 J=1,N
  IF(W2(I,J).GT.(W2(I,Q)+W2(Q,J)+5))GOTO 29
  GOTO 67
  29 W2(I,J)=5+W2(I,Q)+W2(Q,J)
  P(I,J)=P(I,Q)
  67 CONTINUE
  56 CONTINUE
  45 CONTINUE
  DO 77 I=1,N
  DO 88 J=1,N
    X=I
  77 CONTINUE
  88 CONTINUE
  DO 91 M=1,N
    WRITE(26,1000)1,J,W2(I,J)
  1000 FORMAT(I3,' MIN COST OF TRAVEL FROM ORIGIN',*','I3,' TO DESTINATION',*','I3,' IS',*','I6)
  DO 98 M=1,N
F = M + 1
WRITE(26, 2001) ((C2(R(I, J, M), R(I, J, F), K), K = 1, 9), R(I, J, M), R(I, J, F)
*, R1(R(I, J, M), R(I, J, F), 2))
IF(R(I, J, F) .EQ. J) GOTO 92
98 CONTINUE
2001 FORMAT (IX, ' USING LINES', '9', ' FROM ', '9', ' TO ', '9', ' VIA', '9')
92 CONTINUE
91 CONTINUE
STOP
END
Consider a section of the London underground network, (see fig. AP 1). Assume that the time(cost) of travel along any link is 2, and the time(cost) of changing trains(modes) is 5.

Here is a sample of the answer obtained by running either programme Dijkstra 2 or Floyd 2 on the above network.

MIN COST OF TRAVEL FROM ORIGIN 10 TO DESTINATION 30 IS 27
USING LINES 1 0 0 0 0 0 0 0 FROM 10 TO 29 VIA 45
USING LINES 2 0 0 0 0 0 0 0 FROM 29 TO 30 VIA 29

MIN COST OF TRAVEL FROM ORIGIN 10 TO DESTINATION 31 IS 26
USING LINES 1 3 0 0 0 0 0 FROM 10 TO 8 VIA 9
USING LINES 7 0 0 0 0 0 0 FROM 8 TO 38 VIA 37
USING LINES 2 0 0 0 0 0 0 FROM 38 TO 31 VIA 32

MIN COST OF TRAVEL FROM ORIGIN 10 TO DESTINATION 31 IS 26
USING LINES 1 3 0 0 0 0 0 FROM 10 TO 8 VIA 9
USING LINES 7 0 0 0 0 0 0 FROM 8 TO 38 VIA 37
USING LINES 2 0 0 0 0 0 0 FROM 38 TO 31 VIA 32

MIN COST OF TRAVEL FROM ORIGIN 11 TO DESTINATION 46 IS 24
USING LINES 1 3 0 0 0 0 0 FROM 11 TO 8 VIA 9
USING LINES 7 0 0 0 0 0 0 FROM 8 TO 37 VIA 35
USING LINES 6 0 0 0 0 0 0 FROM 37 TO 46 VIA 34

MIN COST OF TRAVEL FROM ORIGIN 11 TO DESTINATION 46 IS 24
USING LINES 1 3 0 0 0 0 0 FROM 11 TO 8 VIA 9
USING LINES 7 0 0 0 0 0 0 FROM 8 TO 37 VIA 35
USING LINES 6 0 0 0 0 0 0 FROM 37 TO 46 VIA 34

MIN COST OF TRAVEL FROM ORIGIN 11 TO DESTINATION 47 IS 23
USING LINES 1 3 0 0 0 0 0 FROM 11 TO 2 VIA 3
USING LINES 6 0 0 0 0 0 0 FROM 2 TO 47 VIA 2

MIN COST OF TRAVEL FROM ORIGIN 19 TO DESTINATION 3 IS 24
USING LINES 4 0 0 0 0 0 0 FROM 19 TO 20 VIA 19
USING LINES 5 0 0 0 0 0 0 FROM 20 TO 4 VIA 34
USING LINES 1 3 0 0 0 0 0 FROM 4 TO 3 VIA 4

MIN COST OF TRAVEL FROM ORIGIN 19 TO DESTINATION 28 IS 18
USING LINES 4 0 0 0 0 0 0 FROM 19 TO 20 VIA 19
USING LINES 7 0 0 0 0 0 0 FROM 20 TO 24 VIA 24

MIN COST OF TRAVEL FROM ORIGIN 20 TO DESTINATION 41 IS 9
USING LINES 6 0 0 0 0 0 0 FROM 20 TO 40 VIA 20
USING LINES 2 0 0 0 0 0 0 FROM 40 TO 41 VIA 40

MIN COST OF TRAVEL FROM ORIGIN 23 TO DESTINATION 31 IS 18
USING LINES 1 9 0 0 0 0 0 FROM 23 TO 24 VIA 23
USING LINES 8 0 0 0 0 0 0 FROM 24 TO 33 VIA 24
USING LINES 2 0 0 0 0 0 0 FROM 33 TO 31 VIA 32
Fig. AP.1

The cost is calculated as the sum of the costs of the segments joining the starting point with the destination, such that

\[ \text{cost} = \sum \text{cost of segments} \]

Note:

The cost is minimal on the selected segment.

Program: Algorithm

This program finds the set of routes \( R \) joining \( s \) to \( t \) such that

\[ \text{cost} = \min \left( \sum \text{cost of segments} \right) \]

where

- \( s \) is the starting point
- \( t \) is the destination
- The cost of each segment \( \text{cost}_{ij} \) is a constant.

\[ \text{cost}_{ij} = \text{distance}_{ij} \]

The cost is calculated as the sum of the costs of the segments joining the starting point with the destination, such that

\[ \text{cost} = \sum \text{cost of segments} \]

Note:

The cost is minimal on the selected segment.

Program: Algorithm

This program finds the set of routes \( R \) joining \( s \) to \( t \) such that

\[ \text{cost} = \min \left( \sum \text{cost of segments} \right) \]

where

- \( s \) is the starting point
- \( t \) is the destination
- The cost of each segment \( \text{cost}_{ij} \) is a constant.

\[ \text{cost}_{ij} = \text{distance}_{ij} \]
Programme: Alpath

This programme finds the set of routes $P$ joining an origin to a destination, such that:

- the cost of travel on route $p < a \times$ the cost of travel on route $q$

where $p$ is a route belonging to $P$
$q$ is the route with the least cost of travel joining the origin-destination pair
$a$ is a constant. In this programme $a = 2$.

Note

The cost of travel on the links is constant.
DIMENSION IW(10,10), IP(100), IRR(10), IG(100), IW1(10,10)
* IO(10,1000,10), IDD(10,1000,10), IMCO(500), IPATH(1000,10),
* IN(10,1000), ICOST(500)
WRITE(6,300)
300 FORMAT('PRINT NUMBER OF NODES IN I2 FORMAT')
READ(5,100)IN
READ(15,110)((IW(I,J),J=1,IN),I=1,IN)
110 FORMAT(8IS)
ICOUNT=0
IFIRST=0
60 DO 44 I=1,IN
 DO 55 J=1,IN
 IW(I,J)=IW(I,J)
55 CONTINUE
44 CONTINUE
73 ITP=0
 DO 17 IM=1,IN
 IP(IM)=0
 IRR(IM)=0
 IG(IM)=0
17 CONTINUE
100 FORMAT(I2)
IPP=0
IT=0
IS=0
 IF(IFIRST.GT.0)GOTO 79
 WRITE(6,200)
111 FORMAT(2I2)
READ(5,111)IR,ID
200 FORMAT('PRINT YOUR ORIGIN IN I2 FORMAT FOLLOWED BY DEST
*INATION')
79 M=10000000
 DO 77 IE=1,IN
 IF(IE.EQ.1)GOTO 70
 DO 10 I=1,IN
 IF(IW1(IR,I).GT.(IW1(IR,IS)+IW1(IS,I)))GOTO 90
10 CONTINUE
 GOTO 70
90 IP(I)=IS
 IW1(IR,I)=(IW1(IR,IS)+IW1(IS,I))
 GOTO 10
70 DO 40 I=1,IN
 IF(I.EQ.IR)GOTO 40
 IF(IW1(IR,I).LT.IT)GOTO 40
 DO 33 IZ=1,IE
 IF(I.EQ.IS)GOTO 40
33 CONTINUE
 IF(IW1(IR,I).LE.M)GOTO 11
40 CONTINUE
 IG(IE)=IS
 IT=M
 IF(IS.EQ.ID)GOTO 98
 M=10000000
 GOTO 77
11 M=IW1(IR,I)
 IS=I
 GOTO 40
77 CONTINUE
IF(IFIRST.GT.0)GOTO 88
IMAX=2*IW1(IR,ID)
IF(IMAX.GT.99999)GOTO 199
88 IF(IW1(IR,ID).GT.IMAX)GOTO 777
151 FORMAT(1X,'A PLAUSIBLE PATH OF TRAVEL FROM ORIGIN*',12,IX,' TO DESTINATION',12,IX,' WITH A TRAVEL COST OF'*
*,14,IX,' IS')
IX=ID
DO 66 I=1,IN
IPP=IN+1-I
IRR(IPP)=IX
IX=IP(IX)
IF(IX.EQ.0)GOTO 23
66 CONTINUE
23 I=I+1
IRR(IN+1-I)=IR
1001 FORMAT(1X,10I8)
L=0
IF(IFIRST.EQ.2)GOTO 57
IFIRST=1
57 DO 121 I=1,IN
IF(IRR(I).GT.O)GOTO 122
L=L+1
121 CONTINUE
ICOUNT=ICOUNT+1
ICOST(ICOUNT)=IW1(IR,ID)
DO 661 J=1,ITP
IPATH(ICOUNT,J)=IRR(J)
661 CONTINUE
GOTO 123
122 IRR(I-L)=IRR(I)
ITP=ITP+1
GOTO 121
123 ITP=ITP-1
IF(IFIRST.EQ.2)GOTO 798
ITP=IPP-1
123 ITP=ITP-1
IF(IFIRST.EQ.2)GOTO 798
IRW=1
ICOD=1
IPIP=1
KLM=ITP
DO 131 K=1,ITP
IO(K,IRCW,1)=IRR(K)
IDD(K,IRW,1)=IRR(K+1)
131 CONTINUE
DO 141 I=1,IN
DO 142 J=1,IN
IW1(I,J)=IW(I,J)
142 CONTINUE
141 CONTINUE
IW1(IO(1,IRROW,ICOD),IDD(1,IRROW,ICOD))=1000000
IFIRST=2
GOTO 73
798 ICOD=1
IRW=IRW+1
IN(IPIP,IRW)=ITP
DO 221 K=1,ITP
IO(IPIP,IRW,K)=IRR(K)
IDD(IPIP,IRW,K)=IRR(K+1)
221 CONTINUE
DO 211 I=1,IN
DO 212 J=1,IN
IW1(I,J)=IW(I,J)
212 CONTINUE
211 CONTINUE
DO 173 II=1,IROW
III=IMCO(II)+1
IW1(II(IPIP, II, III), IDP(IPIP, II, III))=1000000
173 CONTINUE
GOTO 73
777 IF(IROW.EQ.1)GOTO 277
IMCO(IROW)=ICOD
ICOD=ICOD+1
IF(ICOD.GT.IIN(IPIP, IROW))GOTO 177
GOTO 788
177 IROW=IROW-1
IF(IROW.EQ.1)GOTO 277
ICOD=IMCO(IROW)+1
GOTO 777
277 IF(IPIP.EQ.KLM)GOTO 99
IPIP=IPIP+1
ICOD=1
DO 345 I=1, IN
IMCO(I)=0
345 CONTINUE
GOTO 788
199 WRITE(6,7000)IR, ID
7000 FORMAT(IX, 'THERE IS NO PATH LINKING', I2, 'WITH', I2)
99 DO 991 I=1,ICOUNT
IF(ICOST(I).EQ.0)GOTO 991
IF(I.EQ.ICOUNT)GOTO 124
DO 992 J=1,ICOUNT
IF(J.EQ.I)GOTO 992
IF(ICOST(J).EQ.0)GOTO 992
IF(ICOST(I).EQ.ICOST(J))GOTO 911
992 CONTINUE
124 WRITE(16,151)IR, ID, ICOST(I)
WRITE(16,1001)(IPATH(I, LJ), LJ=1, IN)
991 CONTINUE
GOTO 999
999 STOP
END
Consider the network $N(8,24)$ with the following link travel costs:

- $c(1,2) = c(2,1) = 3$
- $c(1,4) = c(4,1) = 5$
- $c(2,3) = 4$
- $c(2,6) = c(6,2) = 7$
- $c(3,6) = c(6,3) = 3$
- $c(3,8) = 4$
- $c(4,5) = c(5,4) = 1$
- $c(4,8) = c(8,4) = 6$
- $c(5,6) = c(6,5) = 4$
- $c(5,8) = 3$
- $c(6,7) = c(7,6) = 7$
- $c(6,8) = c(8,6) = 2$
- $c(7,3) = 3$
- $c(7,8) = c(8,7) = 2$

where $c(a,b)$ is the cost of travel along link $ab$.

To find the set of plausible paths joining any O-D pair, using the algorithm described in chapter 3, carry out the following:

a-call program alpath

b-enter the link travel cost data in the following form

```
0000000039999900005999999999999999999999
000030000000004999999999900007999999999
999999999000099999999990000399999999999
000005999999999900000000199999999999999
9999999999999990000100000000049999999999
99999999999999900000700003999999999999999
99999999999999900003999999999999999999999
99999999999999900006999999999999999999999
```

c-enter the number of nodes followed by O-D pair.

The plausible paths joining O-D pairs $(1,3), (1,8),$ and $(2,8)$ are:

- A plausible path of travel from origin 1 to destination 3 with a travel cost of 7 is:

```
1 2 3 0 0 0 0 0
```

- A plausible path of travel from origin 1 to destination 3 with a travel cost of 13 is:

```
1 4 5 6 3 0 0 0
```

- A plausible path of travel from origin 1 to destination 3 with a travel cost of 14 is:

```
1 4 5 8 7 3 0 0
```

- A plausible path of travel from origin 1 to destination 3 with a travel cost of 14 is:

```
1 4 5 8 6 3 0 0
```

- A plausible path of travel from origin 1 to destination 3 with a travel cost of 13 is:

```
1 2 6 3 0 0 0 0
```
A PLAUSIBLE PATH OF TRAVEL FROM ORIGIN 1 TO DESTINATION 8 WITH A TRAVEL COST OF 9 IS
1 4 5 8 0 0 0 0
A PLAUSIBLE PATH OF TRAVEL FROM ORIGIN 1 TO DESTINATION 8 WITH A TRAVEL COST OF 11 IS
1 2 3 8 0 0 0 0
A PLAUSIBLE PATH OF TRAVEL FROM ORIGIN 1 TO DESTINATION 8 WITH A TRAVEL COST OF 12 IS
1 2 6 8 0 0 0 0
A PLAUSIBLE PATH OF TRAVEL FROM ORIGIN 1 TO DESTINATION 8 WITH A TRAVEL COST OF 17 IS
1 2 6 3 8 0 0 0
A PLAUSIBLE PATH OF TRAVEL FROM ORIGIN 1 TO DESTINATION 8 WITH A TRAVEL COST OF 17 IS
1 2 6 5 8 0 0 0
A PLAUSIBLE PATH OF TRAVEL FROM ORIGIN 1 TO DESTINATION 8 WITH A TRAVEL COST OF 12 IS
1 2 3 6 8 0 0 0
A PLAUSIBLE PATH OF TRAVEL FROM ORIGIN 1 TO DESTINATION 8 WITH A TRAVEL COST OF 17 IS
1 2 3 6 5 8 0 0
A PLAUSIBLE PATH OF TRAVEL FROM ORIGIN 1 TO DESTINATION 8 WITH A TRAVEL COST OF 11 IS
1 4 8 0 0 0 0 0
A PLAUSIBLE PATH OF TRAVEL FROM ORIGIN 1 TO DESTINATION 8 WITH A TRAVEL COST OF 12 IS
1 4 5 6 8 0 0 0
A PLAUSIBLE PATH OF TRAVEL FROM ORIGIN 1 TO DESTINATION 8 WITH A TRAVEL COST OF 17 IS
1 4 5 6 3 8 0 0
A PLAUSIBLE PATH OF TRAVEL FROM ORIGIN 2 TO DESTINATION 8 WITH A TRAVEL COST OF 8 IS
2 3 8 0 0 0 0 0
A PLAUSIBLE PATH OF TRAVEL FROM ORIGIN 2 TO DESTINATION 8 WITH A TRAVEL COST OF 9 IS
2 6 8 0 0 0 0 0
A PLAUSIBLE PATH OF TRAVEL FROM ORIGIN 2 TO DESTINATION 8 WITH A TRAVEL COST OF 12 IS
2 1 4 5 8 0 0 0
A PLAUSIBLE PATH OF TRAVEL FROM ORIGIN 2 TO DESTINATION 8 WITH A TRAVEL COST OF 14 IS
2 1 4 8 0 0 0 0
A PLAUSIBLE PATH OF TRAVEL FROM ORIGIN 2 TO DESTINATION 8 WITH A TRAVEL COST OF 15 IS
2 1 4 5 6 8 0 0
A PLAUSIBLE PATH OF TRAVEL FROM ORIGIN 2 TO DESTINATION 8 WITH A TRAVEL COST OF 14 IS
2 6 3 8 0 0 0 0
A PLAUSIBLE PATH OF TRAVEL FROM ORIGIN 2 TO DESTINATION 8 WITH A TRAVEL COST OF 16 IS
2 6 7 8 0 0 0 0
A PLAUSIBLE PATH OF TRAVEL FROM ORIGIN 2 TO DESTINATION 8 WITH A TRAVEL COST OF 9 IS
2 3 6 8 0 0 0 0
A PLAUSIBLE PATH OF TRAVEL FROM ORIGIN 2 TO DESTINATION 8 WITH A TRAVEL COST OF 16 IS
2 3 6 7 8 0 0 0
Programme: Useopt

This programme finds the user-equilibrium flow distribution for a multi-origin/destination network, where the demand for travel is fixed. As described in section 3.3.
(USEOPT)
IMPLICIT REAL*8(A-H, O-Z)
REAL*8 AA(37, 37), A(50, 50), BB(37), B(50), SIGMA, C1(50), WORK(5000)
*, TTN(8), F(8), TOL
INTEGER S(8, 10), GG(8), PA(8), RR, ZZ, N, M
*, IRANK, T, Z, PP
WRITE(22, 3333)
READ(21, 1100) N, M
WRITE(22, 1500) N, M
READ(21, 1200) (F(I), 1 = 1, M)
WRITE(22, 1201) (F(I), 1 = 1, M)
DO 10 I = 1, N
READ(21, 1000) (A(I, J), J = 1, N)
WRITE(22, 1001) (A(I, J), J = 1, N)
10 CONTINUE
READ(21, 5556) (B(I), I = 1, N)
DO 89 I = 1, N
WRITE(22, 5555) B(I)
89 CONTINUE
WRITE(22, 2222)
READ(21, 2000) (GG(I), I = 1, M)
WRITE(22, 2001) (GG(I), I = 1, M)
READ(21, 3000) (S(J, I), J = 1, M)
WRITE(22, 2001) (S(J, I), J = 1, M)
IF(ISH.EQ.O) GOTO 660
650 DO 66 I = 1, Z
DO 33 J = 1, Z
AA(I, J) = 0.0
33 CONTINUE
66 CONTINUE
660 DO 55 I = 1, Z
BB(I) = 0.0
55 CONTINUE
CALL MAT(S, A, F, M, N, B, Z, T, AA, BB)
WRITE(22, 1122) Z, T
DO 333 I = 1, M
WRITE(22, 2237) (S(I, J), J = 1, 3)
333 CONTINUE
2237 FORMAT(1X, 3I4)
WRITE(22, 1099) (BB(I), I = 1, Z)
DO 31 I = 1, Z
WRITE(22, 1099) (AA(I, J), J = 1, Z)
31 CONTINUE
LWORK = 5000
IFAIL = 0
TOL = 5*0.0001
NRA = 37
CALL F04JDF(Z, T, AA, NRA, BB, TOL, SIGMA, IRANK, WORK
*, LWORK, IFAIL)
WRITE(22, 1122) Z, T
WRITE(22, 2223) S(1, 1), S(2, 1)
WRITE(22, 2223) (S(I, J), J = 1, 3)
WRITE(22, 2223) S(1, 2), S(2, 1), S(2, 2)
IF(ZZ.EQ.1) GOTO 650
DO 11 IP = 1, M
TTN(IP)=1000000.0

11 CONTINUE
DO 12 I=1,M
PA(I)=0
12 CONTINUE
CALL FLPA(M,N,A,GG,TTN,S,BB,Z,PA,B)
WRITE(22,1122)Z,T
WRITE(22,*)S(1,1),S(2,1),S(5,3)
RR=0
CALL AS(M,S,PA,RR)
DO 525 I=1,M
WRITE(22,2237)(S(I,J),J=1,3)
525 CONTINUE
WRITE(22,1122)Z,T
IF(RR.EQ.0)GOTO 600
WRITE(22,661)RR
661 FORMAT(dX, 12)
GOTO 650
600 CALL END(Z,S,BB,N)
WRITE(22,1122)
1100 FORMAT(21)
444 FORMAT(dX,7F5.2)
555 FORMAT(dX,5F10.4)
2221 FORMAT(dX,5I2)
2223 FORMAT(dX,2I2)
1000 FORMAT(27F)
5555 FORMAT(dX,F8.2)
2000 FORMAT(8I)
3000 FORMAT(8I)
1200 FORMAT(8F)
1122 FORMAT(dX,2I2)
3333 FORMAT(dX,´START´)
1500 FORMAT(dX,´NO. OF PATHS = ´,I2,´NO. OF O-D = ´,I2)
1001 FORMAT(dX,27F4.1)
1201 FORMAT(dX,´FLOW BETWEEN EACH O-D = ´,8F6.2)
5556 FORMAT(27F)
1099 FORMAT(dX,27F9.3)
2001 FORMAT(dX,8I4)
2222 FORMAT(dX,´PRINT NO. OF LARGEST PATH BETWEEN EACH O-D IN A VECTOR
*OF I2 FORMAT
*THEN THE VECTOR OF PATHS USED BETWEEN EACH O-D´)
STOP
END
*
************************************************************************************************************
**
************************************************************************************************************
**
SUBROUTINE MAT(S,A,F,M,N,B,Z,T,AA,BB)
IMPLICIT REAL*8(A-H,O-Z)
REAL*8 AA(37,37),A(50,50),B(50),BB(37),F(8)
INTEGER S(8,10),T,Z,R,P,G,M,N,H
Z=0
T=0
J=0
I=0
DO 10 K=1,M
P=I
30 C=S(K,P)
IF(C.EQ.0)GOTO 10
SUBROUTINE PVE(Z, T, M, S, BB, ZZ)
IMPLICIT REAL*8(A-H,0-Z)
REAL*8 BB(37)
INTEGER T, Z, ZZ, K, FF, M, R, S(8,10), GG(8)
IB=1
K=1
FF=0
DO 10 I=1,T
IF(BB(I).GT.0.)GOTO 10
DO 20 R=IB,M
60 IF(S(R,K).EQ.0)GOTO 21
FF=FF+1
IF(FF.EQ.1)GOTO 40
K=K+1
GOTO 60
21 K=1
20 CONTINUE
40 S(R,K)=0
ZZ=1
IB=R
DO 77 II=K,5
ID=II+1
S(R,II)=S(R,ID)
77 CONTINUE
SUBROUTINE FLPA(M,N,A,GG,TTN,S,BB,Z,PA,B)
IMPLICIT REAL*8(A-H,O-Z)
REAL*8 BB(37),A(50,50),C11(50),TTN(8),B(50)
INTEGER PA(8),Z,PP,K,M,N,S(8,10),GG(8)
PP=1
DO 10 K=1,N
L=1
DO 50 J=1,M
1=1
30 IF(S(J,I).EQ.0)GOTO 50
C11(K)=(C11(K)+(A(K,S(J,I))*BB(L)))
L=L+1
1=1+1
GOTO 30
50 CONTINUE
C11(K)=C11(K)-B(K)
IF(K.LE.GG(PP))GOTO 70
PP=PP+1
70 IF(TTN(PP).GT.C11(K))GOTO 20
GOTO 10
20 WRITE(22,2323)
2323 FORMAT(dX,"PPP")
PA(PP)=K
TTN(PP)=C11(K)
10 CONTINUE
DO 77 I=1,N
C11(I)=0.0
77 CONTINUE
RETURN
END

SUBROUTINE AS(M,S,PA,RR)
INTEGER RR,PA(8),M,S(8,10)
DO 10 I=1,M
J=1
30 IF(S(I,J).EQ.PA(I))GOTO 10
IF(S(I,J).EQ.0)GOTO 20
J=J+1
GOTO 30
20 S(I,J)=PA(I)
RR=1
10 CONTINUE
RETURN
END

SUBROUTINE END(Z,S,BB,N)
IMPLICIT REAL*8(A-H,O-Z)
REAL*8 BB(37)
INTEGER Z, S(8,10), R, N
WRITE(22,654)

654 FORMAT(1X, "END")
R = 1
DO 77 I = 1, N
L = 1
30 IF(S(I, L).EQ.0) GOTO 77
WRITE(22,7777)S(I, L), BB(R)
L = L + 1
R = R + 1
GOTO 30
77 CONTINUE
7777 FORMAT(1X, 'NO. OF PASS. USING PATH', I4, ' IS ', F20.15)
RETURN
END

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
Example

Consider a network with 8 origin-destination points and 27 paths (routes) joining them such that:

- O-D 1 is joined by routes 1, 2, 3, 4, 5, 6.
- O-D 2 is joined by routes 7, 8.
- O-D 3 is joined by route 9.
- O-D 4 is joined by routes 10, 11, 12.
- O-D 5 is joined by routes 13, 14, 15, 16, 17, 18, 19.
- O-D 6 is joined by routes 20, 21.
- O-D 7 is joined by routes 22, 23, 24.
- O-D 8 is joined by routes 25, 26, 27.

The cost of travel on each route is as follows:

\[ C(P1) = 13 F_1 + 2 F_2 + 11 F_4 + 11 F_6 + 11 F_{11} + 2 F_{12} + 2 F_{15} + 2 F_{16} + 11 F_{18} + 11 F_{23} + 11 F_{24} + 11 F_{27} + 20. \]

\[ C(P2) = 2 F_1 + 8 F_2 + 3 F_3 + 5 F_4 + 5 F_5 + 5 F_6 + 10 + 2 F_{12} + 5 F_{13} + 5 F_{14} + 2 F_{15} + 2 F_6 + 5 F_7 + 17 + 5 F_{17} + 5 F_{23} + 5 F_{24} + 5 F_{27} + 33. \]

\[ C(P3) = F_2 + 14 F_3 + 13 F_4 + 11 F_5 + 10 F_6 + 13 F_7 + 10 F_8 + 3 F_9 + 3 F_{12} + 10 F_{13} + 3 F_{14} + 10 F_{15} + 10 F_{16} + 10 F_{17} + 10 F_{18} + 10 F_{20} + 26 F_{23} + 29 F_{24} + 26 F_{27} + 21. \]

\[ C(P4) = 11 F_1 + 5 F_2 + 13 F_3 + 29 F_4 + 10 F_5 + 26 F_6 + 13 F_7 + 10 F_8 + 3 F_9 + 5 F_{10} + 11 F_{11} + 3 F_{12} + 15 F_{13} + 5 F_{14} + 3 F_{16} + 15 F_{17} + 11 F_{18} + 10 F_{20} + 26 F_{23} + 29 F_{24} + 26 F_{27} + 21. \]

\[ C(P5) = 1 F_1 + 2 F_2 + 11 F_3 + 10 F_4 + 14 F_5 + 13 F_6 + 10 F_7 + 13 F_8 + 12 + 10 F_{12} + 10 F_{13} + 10 F_{15} + 10 F_{16} + 10 F_{17} + 10 F_{20} + 22 F_{22} + 11 F_{23} + 12 F_{24} + 10 F_{27} + 36. \]

\[ C(P6) = 12 F_1 + 6 F_2 + 10 F_3 + 26 F_4 + 13 F_5 + 29 F_6 + 10 F_7 + 13 F_8 + 5 F_{10} + 11 F_{11} + 12 + 15 F_{13} + 14 F_{14} + 15 F_{15} + 15 F_{16} + 11 F_{17} + 10 F_{19} + 21 F_{21} + 27 F_{22} + 28 F_{23} + 26 F_{24} + 8. \]

\[ C(P7) = 13 F_3 + 13 F_4 + 10 F_5 + 6 + 13 F_6 + 13 F_7 + 10 F_8 + 3 F_9 + 3 F_{12} + 10 F_{13} + 3 F_{16} + 10 F_{17} + 10 F_{20} + 10 F_{23} + 13 F_{24} + 10 F_{27} + 8. \]

\[ C(P8) = 1 F_1 + 2 F_2 + 10 F_3 + 10 F_4 + 13 F_5 + 13 F_6 + 10 F_7 + 13 F_8 + 12 + 10 F_{12} + 10 F_{13} + 15 F_{15} + 16 + 10 F_{17} + 10 F_{20} + 12 F_{22} + 11 F_{23} + 12 F_{24} + 10 F_{27} + 16. \]

\[ C(P9) = 3 F_3 + 3 F_4 + 7 F_5 + 9 F_9 + 5 F_{11} + 14 F_{13} + 15 + 3 F_{16} + 18 F_{17} + 19 F_{20} + F_{24} + 2 F_{24} + 5 F_{25} + 7 F_{26}. \]

\[ C(P10) = 5 F_{22} + 5 F_4 + 5 F_6 + 6 + 10 + 5 F_{13} + 6 F_{14} + 5 F_{15} + 25 F_{21} + 5 F_{23} + 5 F_{24} + 5 F_{27} + 8. \]

\[ C(P11) = 11 F_1 + 2 F_2 + 11 F_3 + 11 F_4 + 5 F_5 + 11 F_6 + 13 F_{11} + 13 F_{18} + 2 F_{22} + 11 F_{23} + 11 F_{24} + 2 F_{25} + 11 F_{27} + 14. \]

\[ C(P12) = 2 F_{1} + 2 F_{2} + 3 F_4 + 3 F_7 + 7 F_9 + 9 F_{12} + 5 F_{13} + 14 F_{14} + 2 F_{15} + 5 F_{16} + 18 F_{19} + 4 F_{20} + 4 F_{24} + 2 F_{24} + 5 F_4 + 25 F_{27} + 16. \]

\[ C(P13) = 5 F_{2} + 10 F_3 + 15 F_4 + 10 F_5 + 15 F_6 + 10 F_7 + 10 F_8 + 6 F_9 + 5 F_{10} + 6 F_{12} + 25 F_{13} + 9 F_{14} + 15 F_{17} + 4 F_{18} + 10 F_{19} + 22 F_{20} + 6 F_{22} + 15 F_{23} + 15 F_{24} + 4 F_{25} + 25 F_{27} + 17. \]
The demand for travel between each O-D pair is:

<table>
<thead>
<tr>
<th>O-D</th>
<th>Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50.00</td>
</tr>
<tr>
<td>2</td>
<td>50.00</td>
</tr>
<tr>
<td>3</td>
<td>20.00</td>
</tr>
<tr>
<td>4</td>
<td>100.00</td>
</tr>
<tr>
<td>5</td>
<td>60.00</td>
</tr>
<tr>
<td>6</td>
<td>100.00</td>
</tr>
<tr>
<td>7</td>
<td>80.00</td>
</tr>
<tr>
<td>8</td>
<td>100.00</td>
</tr>
</tbody>
</table>

end
Using programme USEOPT, the user-equilibrium flow distribution to the above problem is:

<table>
<thead>
<tr>
<th>NO. OF PATHS</th>
<th>NO. OF O-D</th>
<th>FLOW BETWEEN EACH O-D</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>8</td>
<td>50.00 50.00 20.00 100.00 80.00 100.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NO. OF PASS. USING PATH</th>
<th>IS</th>
<th>FLOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>44.276113951789619</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>1.872169466764065</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>3.851716581446306</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>48.624056488921359</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>1.375943511078639</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>19.999999999999991</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>96.815924032140250</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>3.184075967859759</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>59.999999999999999</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>100.000000000000009</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>80.000000000000002</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>100.000000000000000</td>
</tr>
</tbody>
</table>
Programme: Detree

This programme finds the set of complements of trees appearing in $\Delta$. As described in section 4.7.1 "Method 1".
* THIS PROGRAM FINDS THE SET OF TREES OR OTHERWISE APPEARING IN
* THE DENOMINATOR.
* READ THE NO. OF CYCLES AND TOTAL NO. OF LINKS FOLLOWED
* BY THE LINKS FORMING EACH CYCLE FROM FILE 11
INTEGER SIZ, CY_SIZ(20), LINK, II(20), POS_TR(7), NUM, TO_LIN,
* M, N, P, X, Y, TR_LINK(20), TOT_TR, A, B, P_TREE(1600, 20), TR_LIN(20),
* NOT_TR, NU_TRE, CYCLE(7, 20), G, KK(20)
* READING THE DATA
READ(11, *) NUM, TO_LIN
DO 10 I = 1, NUM
  DO 20 J = 1, 20
    READ(11, *) LINK
    IF (LINK .EQ. 0) GOTO 30
    WRITE(22, *) LINK
    CYCLE(I, J) = LINK
    SIZ = SIZ + 1
  CONTINUE
30    CY_SIZ(I) = SIZ
      SIZ = 0
10 CONTINUE
* PRODUCE POS_TR WHICH CONTAINS ONE LINK FROM EACH
* CYCLE AND CONTAINS NO REPEATED LINKS
DO 40 I = 1, CY_SIZ(1)
  II(1) = II(1) + 1
  II(2) = 0
  DO 50 J = 1, CY_SIZ(2)
    II(2) = II(2) + 1
    II(3) = 0
  DO 60 K = 1, CY_SIZ(3)
    II(3) = II(3) + 1
    II(4) = 0
  DO 70 L = 1, CY_SIZ(4)
    II(4) = II(4) + 1
    II(5) = 0
  DO 80 M = 1, CY_SIZ(5)
    II(5) = II(5) + 1
    II(6) = 0
  DO 90 N = 1, CY_SIZ(6)
    II(6) = II(6) + 1
    II(7) = 0
  IF (NUM .EQ. 6) GOTO 11
  DO 100 P = 1, CY_SIZ(7)
    II(7) = II(7) + 1
11   DO 110 Z = 1, NUM
      POS_TR(Z) = 0
110  CONTINUE
DO 120 X = 1, NUM
  DO 130 Y = 1, NUM
    IF (POS_TR(Y).EQ.0) GOTO 140
    IF (CYCLE(X, II(X)).EQ.POS_TR(Y)) GOTO 150
130  CONTINUE
150 IF (X.EQ.1) GOTO 40
    IF (X.EQ.2) GOTO 50
    IF (X.EQ.3) GOTO 60
    IF (X.EQ.4) GOTO 70
    IF (X.EQ.5) GOTO 80
    IF (X.EQ.6) GOTO 90
IF(NUM.EQ.6)GOTO 140
IF(X.EQ.7)GOTO 100
140 POS_TR(Y)=CYCLE(X,II(X))
120 CONTINUE
* NOW CONVERT THE UNORDERED POS_TR TO P_TREE
* WHOSE LINKS ARE IN AN INCREASING ORDER THIS
* IS DONE BY USING A 0,1 VECTOR TR_LIN
DO 155 G=1,TO_LIN
TR_LIN(G)=0
155 CONTINUE
DO 156 IG=1,NUM
TR_LIN(POS_TR(IG))=1
156 CONTINUE
IL=0
TOT_TR=TOT_TR+1
DO 201 JL=1,20
IF(TR_LIN(JL).EQ.0)GOTO 201
IL=IL+1
P_TREE(TOT_TR,IL)=JL
201 CONTINUE
IF(NUM.EQ.6)GOTO 90
100 CONTINUE
90 CONTINUE
80 CONTINUE
70 CONTINUE
60 CONTINUE
50 CONTINUE
40 CONTINUE
* NOW REMOVE ALL P_TREE THAT ARE REPEATED AN EVEN NO. OF TIMES
LL=TOT_TR-1
DO 210 I=1,LL
M=I+1
DO 220 J=M,TOT_TR
IF(I.EQ.J)GOTO 220
IF(P_TREE(I,1).EQ.99)GOTO 210
IF(P_TREE(J,1).EQ.99)GOTO 220
DO 230 IM=1,TO_LIN
IF(P_TREE(I,IM).EQ.P_TREE(J,IM))GOTO 230
GOTO 220
230 CONTINUE
P_TREE(I,1)=99
NOT_TR=NOT_TR+2
P_TREE(J,1)=99
GOTO 210
220 CONTINUE
210 CONTINUE
NU_TRE=TOT_TR-NOT_TR
DO 301 J=1,TOT_TR
IF(P_TREE(J,1).EQ.99)GOTO 301
WRITE(22,5555)(P_TREE(J,K),K=1,NUM)
301 CONTINUE
5555 FORMAT(1X,6I3)
GOTO 88
88 WRITE(22,3001)NU_TRE
3000 FORMAT(1X,16I3)
3001 FORMAT(1X,'THE TOTAL NUMBER OF TREES IS ',I5)
STOP
END
Consider the network $N(9,16)$ with four 0-D pairs, namely
$01-D_1$, $01-D_2$, $03-D_3$ and $02-D_2$, shown in (Fig. AP 2). Let
the set of cycles appearing in the numerator be:

- $a- 3, 4, 8, 10$
- $b- 1, 2, 4, 5$
- $c- 6, 7, 11, 12$
- $d- 9, 11, 13, 14$
- $e- 6, 10, 15$
- $f- 8, 13, 16$.

Using programme DETREE we can obtain the set of complements
of trees require to find $\triangle$. A sample of this set is:

\[
\begin{align*}
1 & 3 & 6 & 9 & 10 & 13 \\
1 & 3 & 6 & 8 & 9 & 15 \\
1 & 3 & 6 & 9 & 13 & 15 \\
1 & 3 & 7 & 9 & 10 & 13 \\
1 & 3 & 7 & 9 & 10 & 16 \\
1 & 3 & 7 & 8 & 10 & 14 \\
1 & 3 & 7 & 10 & 13 & 14 \\
2 & 6 & 8 & 10 & 14 & 15 \\
2 & 6 & 7 & 8 & 10 & 11 \\
2 & 7 & 10 & 13 & 14 & 15 \\
4 & 6 & 8 & 11 & 14 & 16 \\
4 & 8 & 11 & 13 & 14 & 15 \\
4 & 8 & 9 & 12 & 15 & 16 \\
5 & 6 & 8 & 10 & 14 & 15 \\
5 & 6 & 10 & 13 & 14 & 15 \\
5 & 6 & 10 & 14 & 15 & 16 \\
5 & 6 & 7 & 9 & 10 & 16 \\
5 & 7 & 8 & 9 & 10 & 15 \\
5 & 7 & 9 & 10 & 13 & 15 \\
5 & 7 & 9 & 10 & 15 & 16 \\
5 & 6 & 7 & 8 & 10 & 11 \\
5 & 6 & 7 & 10 & 11 & 16 \\
5 & 7 & 8 & 10 & 11 & 15 \\
5 & 7 & 10 & 11 & 13 & 15 \\
5 & 7 & 10 & 11 & 15 & 16 \\
5 & 7 & 8 & 10 & 13 & 15 \\
5 & 7 & 10 & 13 & 15 & 16 \\
5 & 6 & 7 & 14 & 10 & 15 \\
5 & 6 & 7 & 14 & 15 & 16 \\
5 & 7 & 8 & 10 & 14 & 16 \\
5 & 7 & 8 & 10 & 14 & 15 \\
5 & 7 & 8 & 10 & 13 & 14 \\
5 & 7 & 8 & 10 & 14 & 16 \\
5 & 7 & 8 & 10 & 14 & 15 \\
5 & 7 & 13 & 14 & 15 & 16 \\
5 & 7 & 14 & 15 & 16 & 17 \\
5 & 6 & 8 & 9 & 10 & 11 \\
5 & 6 & 9 & 10 & 11 & 13 \\
5 & 9 & 10 & 11 & 15 & 16 \\
5 & 6 & 10 & 11 & 13 & 16 \\
5 & 6 & 10 & 11 & 13 & 15 \\
5 & 10 & 11 & 13 & 15 & 16 \\
5 & 6 & 8 & 10 & 11 & 14
\end{align*}
\]

(Fig. AP 2)
This programme finds the set of pseudo-routes joining any origin-destination pair of a transportation network. The set of pseudo-routes is used in evaluating $\Delta_{q}$ as described in section 4.7.1 following theorem 4.7.1.
Read the total number of links in the graph; \( Z \)
Read the total number of known pseudo-cycles in the graph; \( C_y \)
Read the maximum number of links in the pseudo-routes
Read the number of routes joining o-d pair \( w;k \)
Read the set of used routes \( p(i,j) \) as a 0,1,-1 matrix and store them in a stack
Read the set of known cycles \( c(i,j) \) as a 0,1,-1 matrix
((Every cycle is read twice: clockwise and anti-clockwise))

**SUB. PRINT**
Print the set of routes in the stack

**SUB. CLEAN**
Remove all routes that contain a cycle from the stack

**SUB. COMMON**
Route \( p(num,) \) and cycle \( c(i,j) \) have a link in common

**SUB. CREATE**
Add route \( p(num,) \) to cycle \( c(i,j) \), creating a possible route \( Ip \)

**SUB. CLEAR**
Discard \( Ip \)

**SUB. LENGTH**
The number of links in \( Ip > n \)

**SUB. GENUINE**
Check the authenticity of \( Ip \) and the routes in the stack

For \( p=1 \) to \( k \)

Add \( Ip \) to stack

Every link in \( p \) belongs to \( Ip \)

Every link in \( Ip \) belongs to \( p \)

\( Ip \) contains a cycle

\( Ip \) is a subset of \( p \)

Remove \( p \) from the stack; \( k-1 \)

Every link in \( p \) belongs to \( Ip \)

Mark \( Ip \)

Next \( p; p+1 \)

Remove \( p \) from the stack
this prog will find the paths for an o-d pair as produced in the
numerator taking into account the sign
integer i,ii,j,p(10000,16),k,n,cy,c(15,16),lp(16),t,num,cl,z,r,
*un_tru(10000),mm,zz,nn,S,M,iit,ff
call readd(z,k,c,n,cy,p)
mm=1
do 10 num=1,1300
if(num.gt.k)goto 500
do 20 cl=l,cy
if(p(num,l).eq.99)goto 10
call common(r,p,c,z,num,cl)
if(r.eq.0) goto 220
call create(z,lp,p,num,c,cl)
call lenthchq(lp,n,z,iit)
if(iit.eq.1) goto 220
call genuine(t,k,z,p,lp,c,cy,un_tru,mm)
if(t.eq.1)goto 220
k=k+l
call store(lp,k,z,p)
220 call clear(lp,z)
20 continue
10 continue
500 call clean(p,k,un_tru,mm,z)
call print(p,z,k)
stop
end
* *****************************************************
* *****************************************************
subroutine readd(z,k,c,n,cy,p)
integer ii,j,z,k,c(15,16),n,cy,p(10000,16)
* this sub will read data from file 11
* z is total no. of links in graph
* n is max no. of links in a path=no. of links in tree
* k is no. of paths for o_d i
* cy is no. of cycles in o_d j for all j
read(11,*)z,k,cy,n
write(22,1111)z,k,cy,n
1111 format(1x,4i2)
do 30 ii=1,k
read(11,*)p(ii,j),j=1,z
write(22,444)(p(ii,j),j=1,z)
30 continue
do 150 ii=1,cy
read(11,*)c(ii,j),j=1,z
WRITE(22,444)(C(ii,J),J=1,Z)
150 continue
444 format(1x,15i2)
return
end
* *****************************************************
* *****************************************************
subroutine lenthchq(lp,n,z,iit)
integer lp(16),n,z,iit,j
* check the no. of links in lp
iit=0
j=0
do 50 i=1,z
if(lp(i).eq.0)goto 50
j=j+1
50  continue
if(j.ge.n)goto 55
iit=1
55  return
end
***************
***************
subroutine genuine(t,k,z,p,lp,c,cy,un_tru,mm)
integer t,k,z,p(10000,16),lp(16),un_tru(10000),mm,s,nn,
cy,c(15,16),ich,jf,im
* find if some path p is a subset of path Ip
* then if Ip is a subset of some path p
* t=0
do 60  j=1,10000
if(j.gt.k)goto 22
if(p(j,1).eq.0)goto 60
* check if lp>=some p
* if yes goto do 71 loop to check the =
do 70  i=1,z
if(p(j,i).eq.0)goto 70
if(lp(i).eq.0)goto 87
70  continue
do 71  i=1,z
if(lp(i).eq.0)goto 71
if(p(j,i).eq.0)goto 23
71  continue
* at this stage Ip and p have the same links so we
* check the sign and value of those elements(links)
iij=0
iik=0
do 67  i=1,z
IKK=LP(I)*P(J,I)
IF(C(IJK).LT.0)GOTO 60
if(lp(i).eq.p(j,i))goto 67
if(lp(i).lt.0)goto 68
if(p(j,i).gt.lp(i))goto 69
goto 79
68  if(p(j,i).lt.lp(i))goto 69
goto 79
69  iij=1
goto 67
79  iik=1
67  continue
iii=iij+iik
if(iii.eq.2)goto 60
if(iii.eq.1)goto 27
t=1
goto 22
27  p(j,1)=-99
GOTO 60
* at this stage p<lp so we check if Ip has a cycle
23  do 81  i=1,cy
do 82  jk=1,z
if(c(i,jk).eq.0)goto 82
nn=lp(jk)*c(i,jk)
if(nn.le.0)goto 81
82  continue
t=1
goto 22

continue

un_tru(mm)=k+1
mm:mm+1
goto 60

do 77 i=1,z
if(lp(i).eq.0)goto 77
if(p(j,i).eq.0)goto 60

continue

do 78 im=1, cy
do 89 jf=1,z
if(c(im,jf).eq.0)goto 89
ich=p(j,jf)*c(im,jf)
if(ich.le.O)goto 78

continue

p(j,1)=99

continue

22 return

end

*           ***************
*           ******************
subroutine create(z,lp,p,num,c,cl)
integer z,lp(16),p(10000,16),num,c(15,16),cl
* add a cycle to a path

do 80 i=1,z
lp(i)=p(num,i)+c(cl,i)

80 continue
return

end

*           ***************
*           ******************
subroutine commonCr,p,c,z,num,cl)
integer r,p(10000,16),c(15,16),z,num,cl
* check if path(num) has a link in common with cycle(cl)
r=0

do 90 i=1,z
if(p(num,i).eq.0)goto 90
if(c(cl,i).eq.0)goto 90
r=1

goto 320

90 continue

320 return

end

*           ***************
*           ******************
subroutine store(lp,k,z,p)
integer lp(16),k, z,p(10000,16)
* hang the newly created path at end of path list

do 330 i=1,z
p(k,i)=lp(i)

330 continue
return

end

*           ***************
*           ******************
subroutine clear(lp,z)
integer lp(16),z
* clear path lp=let lp=(0,0,0...)
do 3150 i=1,z
  lp(i)=0
3150 continue
return
end
*
* *************************************************
* *************************************************
subroutine clean(p,k,un_tru,mm,z)
integer p(10000,16),k,z,n,m,mm,un_tru(10000),ff,zz
* this sub. removes all paths that have any cycle
* in it (particularly those cycles that do not belong to c)
do 111 zz=1,mm
  ff=un_tru(zz)
do 222 j=1,k
  do 333 i=1,z
    if(p(j,1).eq.99)goto 222
    if(j.eq.ff)goto 222
    if(p(j,i).eq.0)goto 333
    if(p(j,i).gt.0)goto 433
    if(p(ff,i).le.p(j,i))goto 333
    goto 222
  433 if(p(ff,i).ge.p(j,i))goto 333
  goto 222
  333 continue
  p(ff,1)=99
  goto 111
222 continue
111 continue
return
end
*
* *************************************************
* *************************************************
subroutine print(p,z,k)
integer pClOOOO,16),z,k
do 350 i= 1,k
  if(p(i,1).eq.99)goto 350
  write(22,600)(p(i,j),j=1,z)
350 continue
write(22,2323)k
2323 format(1x,i? )
600 format(1x,l5i3)
return
end
*
Consider network $N(8,15)$ with four O-D pairs (O1-D1, O4-D4, O2-D2 and O3-D3) and six cycles \{(1,3,4,5),(5,7,10,11), (2,3,4,6),(6,7,10,12),(6,8,10,12,13,15) and (7,9,14,15)\}, see (fig AP 3).

To find the set of pseudo-routes joining origin-destination pair O1-D1, carry out the following:

1- Call programme NNMRTP 1.
2- Input the set of cycles together with a route joining O1-D1 (say 1,4,10).
3- Run the programme.

The pseudo-routes are:

<table>
<thead>
<tr>
<th>link</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

(Fig. AP 3)
2 1 0 1 -1 0 0 0 -1 0 0 -1 0 1 1
0 -1 0 1 -1 0 1 0 0 0 2 1 0 0 0
0 -1 0 1 0 1 0 1 0 0 1 0 -1 0 1
0 -1 0 1 -1 0 0 0 -1 0 2 1 0 1 1
0 -1 0 1 -1 0 0 1 0 0 2 1 -1 0 1


Foulds L. (1985). Traffic network design with user-predictive
REFERENCES


Fisk C. and Nguyen S.(1979) Properties of the two mode equilibrium assignment model with flow dependent transit costs. Centre de research sur les Transports _ publication 121.


Foulds L.(1985) Traffic network design with turn prohibition


Smith M.J. (1978) In a road network, increasing delay locally can reduce delay globally. Transp. Res. 12, 419-422.


Wilkie D.F. and Stefanek G. (1971) Prec ise determination of
equilibrium in travel forecasting problems using numerical optimization techniques. Highway Research Record, 369, 239-252.
