REPORT NO. DOT-TSC-RSPD-78-8,II

NETWORK AGGREGATION IN TRANSPORTATION PLANNING
Volume II: A Fixed Point Method for Treating Traffic Equilibria

Harold W. Kuhn

Mathtech, Inc.
P.O. Box 2392
Princeton NJ 08540

APRIL 1978

FINAL REPORT

DOCUMENT IS AVAILABLE TO THE U.S. PUBLIC THROUGH THE NATIONAL TECHNICAL INFORMATION SERVICE, SPRINGFIELD, VIRGINIA 22161

Prepared for

U.S. DEPARTMENT OF TRANSPORTATION

RESEARCH AND SPECIAL PROGRAMS DIRECTORATE

Office of Transportation Programs Bureau

Office of Systems Engineering
Washington DC 20590

NOTICE

This document is disseminated under the sponsorship of the Department of Transportation in the interest of information exchange. The United States Government assumes no liability for its contents or use thereof.

NOTICE

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the object of this report.

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT-TSC-RSPD-78-8, II		

4. Title and Subtitle
NETWORK AGGREGATION IN TRANSPORTATION PLANNING
Volume II: A Fixed Point Method for Treating Traffic Equilibria

5. Report Date
April 1978

6. Performing Organization Code

7. Author(s)
Harold W. Kuhn

8. Performing Organization Report No.
DOT-TSC-RSPD-78-8,II

9. Performing Organization Name and Address
Mathtech, Inc.*
P.O. Box 2392
Princeton NJ 08540

10. Work Unit No. (TRAIS)
OS850/R8526

11. Contract or Grant No.
DOT-TSC-1232-2

12. Sponsoring Agency Name and Address
U.S. Department of Transportation
Research and Special Programs Directorate
Office of Transportation Programs Bureau
Office of Systems Engineering
Washington DC 20590

13. Type of Report and Period Covered
FINAL REPORT
July 1976-July 1977

14. Sponsoring Agency Code

15. Supplementary Notes
*Under contract to: U.S. Department of Transportation
Transportation Systems Center
Kendall Square
Cambridge MA 02142

16. Abstract

This, the second of two volumes, defines a new algorithm for the network equilibrium model that works in the space of path flows and is based on the theory of fixed point method.

Volume I summarizes research on network aggregation in transportation models.

Volume I has 100 pages.

17. Key Words	18. Distribution Statement
Aggregation Duality Networks Traffic Assignment Fixed Point Method	DOCUMENT IS AVAILABLE TO THE U.S. PUBLIC THROUGH THE NATIONAL TECHNICAL INFORMATION SERVICE, SPRINGFIELD, VIRGINIA 22161

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	76	

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

Preface

This report summarizes the research on Aggregation in Transportation Networks conducted by MATHTECH, Inc. under contract DOT-TSC-1232 for the U.S. Department of Transportation. Funding for the work was provided under Project TARP (OST/TST) with additional support provided by UMTA. The goals of this study were broadly defined as the identification of aggregation practices and the development of a framework for studying these practices. These goals have been accomplished by, (a) conducting a survey of aggregation practices, (b) formulating an aggregation model, (c) conducting a computational study, (d) deriving mathematical programming formulations aimed at making steps of the model precise, and (e) programming and testing a particular algorithm proposed in earlier MATHTECH research.

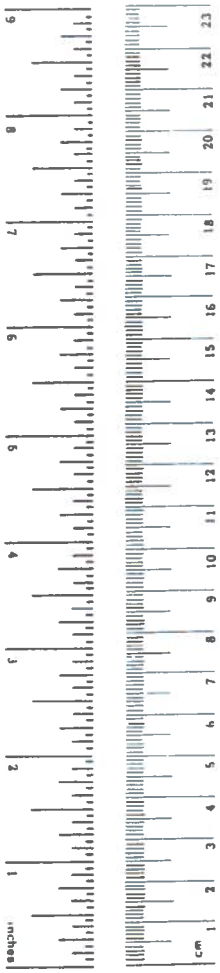
A number of people were very helpful in the survey stage of this project. In particular, thanks are due to William S. Mann, Washington Council of Governments Transportation Planning Board, Morris J. Rothenberg, JHK Associates, Bob Dial of UMTA, Raphael Kedar and Tom Bouvé of FRA for the time taken to discuss their aggregation schemes.

Principal consultant on this project was Professor Harold W. Kuhn, Princeton University, who offered many helpful suggestions throughout the study. A number of the ideas explored here originated in the MATHTECH report "Aggregation in Net work Models for Transportation Planning" (DOT-TSC-883) by Harold W. Kuhn and Daniel E. Cullen. The research was guided throughout by Dr. Edwin J. Roberts and Mr. Michael Nienhaus of TSC and by Mr. Robert Crosby of the Office of the Secretary.

METRIC CONVERSION FACTORS

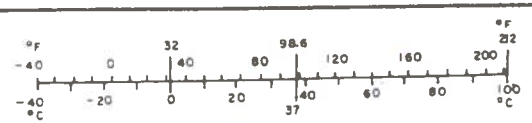
Approximate Conversions to Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
		LENGTH		
in	inches	2.5	centimeters	cm
ft	feet	30	centimeters	cm
yd	yards	0.9	meters	m
mi	miles	1.6	kilometers	km
		AREA		
in^2	square inches	6.5	square centimeters	cm^2
ft^2	square feet	0.09	square meters	m^2
yd^2	square yards	0.8	square meters	m^2
mi^2	square miles	2.6	square kilometers	km^2
	acres	0.4	hectares	ha
		MASS (weight)		
oz	ounces	28	grams	g
lb	pounds	0.45	kilograms	kg
	short tons (2000 lb)	0.9	tonnes	t
		VOLUME		
tsp	teaspoons	5	milliliters	ml
Tbsp	tablespoons	15	milliliters	ml
fl oz	fluid ounces	30	milliliters	ml
c	cups	0.24	liters	l
pt	pints	0.47	liters	l
qt	quarts	0.95	liters	l
gal	gallons	3.8	liters	l
ft^3	cubic feet	0.03	cubic meters	m^3
yd^3	cubic yards	0.76	cubic meters	m^3
		TEMPERATURE (exact)		
°F	Fahrenheit temperature	5/9 (after subtracting 32)	Celsius temperature	°C



Approximate Conversions from Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
		LENGTH		
mm	millimeters	0.04	inches	in
cm	centimeters	0.4	inches	in
m	meters	3.3	feet	ft
m	meters	1.1	yards	yd
km	kilometers	0.6	miles	mi
		AREA		
cm^2	square centimeters	0.16	square inches	in^2
m^2	square meters	1.2	square yards	yd^2
km^2	square kilometers	0.4	square miles	mi^2
ha	hectares (10,000 m^2)	2.5	acres	
		MASS (weight)		
g	grams	0.035	ounces	oz
kg	kilograms	2.2	pounds	lb
t	tonnes (1000 kg)	1.1	short tons	
		VOLUME		
ml	milliliters	0.03	fluid ounces	fl oz
l	liters	2.1	pints	pt
l	liters	1.06	quarts	qt
l	liters	0.26	gallons	gal
m^3	cubic meters	35	cubic feet	ft^3
m^3	cubic meters	1.3	cubic yards	yd^3
		TEMPERATURE (exact)		
°C	Celsius temperature	9/5 (then add 32)	Fahrenheit temperature	°F



CONTENTS

PATHFIX: A FIXED POINT METHOD FOR COMPUTING TRAFFIC EQUILIBRIA (THEORY)

1. Introduction

This report presents the theory of a new algorithm for the network equilibrium model that works in the space of path flows using a label and pivot technique. The report that follows documents a computer program that implements this theory. Together, these two reports provide a detailed elaboration of the proposal advanced by the present author in the MATHTECH report "Aggregation in Network Models for Transportation Planning: Paper 3," (October, 1975) prepared for D.O.T. under contract DOT-TSC-883.

In this introduction we shall attempt to explain the relation of this algorithm to other similar algorithms that have been used for the computation of economic equilibria. The terms "label and pivot" have been used in computing economic equilibria since the initial work of Scarf in this area ten years ago [1]. Since then, the methods have been extended by a number of other mathematicians and economists. (A comprehensive survey by Kuhn [2] has a bibliography of 42 items in print in 1974.)

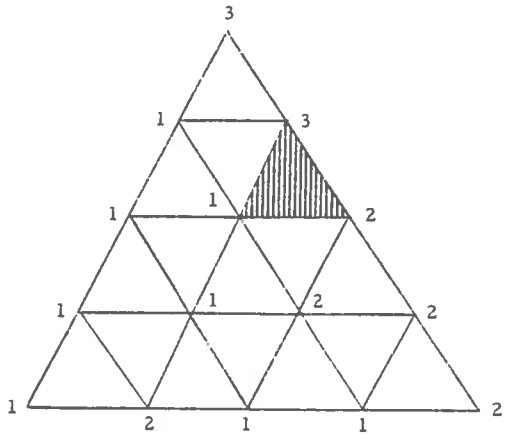
The idea of a pivotal method that is extended to the traffic assignment problem in this paper can be motivated by a classical model of equilibrium in an exchange economy. In such an economy, goods $k = 1, \ldots, n$ are traded at prices $p_1 \ldots, p_n$. A price vector $p = (p_1, \ldots, p_n)$ generates demand and supply for each of these goods; their difference is called the "excess demand function" $g_k(p)$ for the good k. These n functions are assumed to be continuous and homogeneous of degree zero for nonnegative,

nonzero prices. Also, the prices and excess demands are assumed to satisfy Walras' Law which says that expenditures equal revenues when summed over all goods, that is,

$$p_1 g_1(p) + \ldots + p_n g_n(p) = 0 \quad \text{for all} \quad p \geqq 0.$$

Equilibrium for such a model is a set of prices $\bar{p}$ that generate a supply that is not less then the demand for each good, that is, such that $g_k(\bar{p}) \leqq 0$ for $k = 1, \ldots, n$. (Of course, Walras' Law implies that, in equilibrium, the price of a good with positive excess supply is zero.)

The algorithm works on the price simplex where nonnegative, nonzero prices are normalized to sum to one and starts with the standard subdivision of this simplex (see [3]). The vertices of this subdivision are given labels by the following rule: A price vector p is given the label $k \in \{1, \ldots, n\}$ if the excess demand for good k is smallest among those goods with positive price. (If there are ties, take any specific eligible index, for instance, the smallest.) A possible labelling for a case of 3 goods is shown below. Note that the vertices must carry the index of the only good with positive price while the points along an edge must carry one of the two labels of the endpoints.



Of course, Walras' Law implies:

(*) If the label of p is k then $g_k(p) \leqq 0$.

Furthermore, the rule for labelling implies:

(**) If the label of p is k then $p_k > 0$.

Property (**) is the single hypothesis of Sperner's Lemma [4], which asserts the existence of at least one piece of the subdivision with all labels. (In the figure above, the single such triangle is shaded.)

This is enough to approximate an equilibrium arbitrarily closely. For a fine enough subdivision, the points in a completely labelled piece satisfy

$$g_k(p) \leqq \epsilon \qquad \text{for given } \epsilon > 0 \text{ and } k = 1, \ldots, n.$$

Hence, they are nearly equilibria. As the subdivision is refined, we can choose a subsequence of completely labelled pieces that converges to an equilibrium.

In the algorithm that follows, the ideas contained in this simple illustration are generalized and extended in the following ways:

(1) We work not on a price simplex but in the space of proportionate flows on paths joining origin-destination pairs. Geometrically, this space is the Cartesian product of simplices.

(2) This product set is subdivided in an efficient way for computer calculation.

(3) An appropriate labelling that produces a traffic assignment equilibrium is constructed.

(4) The technique of "restarting" [5] to produce extremely accurate answers is generalized to cover this case.

The algorithm is intended to apply to networks for which an extremely accurate answer for the equilibrium is desired. Furthermore, the algorithm

presumes an aggregation by extraction of paths. In common sense terms, it is presumed that there are only a small number of candidate paths joining each origin-destination pair. However, it should be noted that, although the intended application is to an extracted set of paths, if all paths are used then the algorithm constructs (in the limit) an exact equilibrium. Also the extracted subset of paths can be modified easily on the basis of intermediate calculations. For example, we may add a path if it is definitely cheaper than those in use or delete a path if it has nearly zero flow at the current approximation. The restart technique uses the available approximation in an efficient and flexible manner.

The remainder of this paper is organized as follows. In Section 2, essential notation is introduced and the equilibria of traffic flow are defined. In Section 3, two labelling rules are presented and explored combinatorially. The relation between the labelling rules and equilibria is established in Section 4. The material of Section 5 deals with subdivision and restarting. The algorithm that results is assembled in Section 6. A final section, Section 7, treats a small example.

2. Equilibria for Traffic Flow in a Network

Implicitly, the analysis of traffic flow uses a network of directed links, the one-way sections of road used in some appropriate modelling of a real road network. This directed graph will motivate our definitions but will not appear explicitly. Certain nodes in this network are considered to originate traffic (or trips) ending in other nodes called destinations. These pairs (origin-destination or OD) are assumed to be p in number and indexed by k.

OD pairs: $k = 1, \ldots, p$.

The demand for traffic on OD pair k is assumed to be a positive number $d_k > 0$. This may, in an application, be the number of trips to be made from the origin to the destination in a fixed time period.

Demand (trips) on OD pair k: $d_k > 0$, $k = 1, \ldots, p$.

The trips joining OD pair are assumed to take place on a finite number $n_k \geq 2$ of paths which are indexed by $j = 1, \ldots, n_k$.

Path joining OD pair k: $j = 1, \ldots, n_k$ where $n_k \geq 2$.

The paths are composed of (directed) links in the underlying network. There are a finite number q of links used to make the paths considered; these links are indexed by $\ell = 1, \ldots, q$.

Links: $\ell = 1 \ldots, q$.

The occurrence of links on a path is specified by link-path incidence matrices. For OD pair k, such a matrix has q rows and n_k columns; a 1 indicates the presence of a link on the path and a 0 indicates its absence.

Link-path incidence matrices: $\left\{A^{(1)}, \ldots, A^{(p)}\right\}$

$$A^{(k)} = (a_{\ell j}^{(k)}) \text{ for } \ell = 1, \ldots, q;\ j = 1, \ldots, n_k;\ k = 1, \ldots, p$$

and $$a_{\ell j}^{(k)} = \begin{cases} 1 & \text{if link } \ell \text{ is on path } j \text{ joining OD pair } k \\ 0 & \text{otherwise.} \end{cases}$$

For notational convenience, we shall work with the proportions of traffic flow on the paths joining OD pair k. These will be nonnegative vectors of dimension n_k summing to one for each OD pair $k = 1, \ldots, p$. For OD pair k, the proportionate flows form a simplex S_{n_k} of dimension $n_k - 1$.

Proportionate flows on paths joining OD pair k:

$$X_k = (x_{k1}, \ldots, x_{kj}, \ldots, x_{kn_p}) \in S_{n_k} = \left\{(x_{kj}) \mid x_{kj} \geq 0, \textstyle\sum_j x_{kj} = 1\right\}.$$

Given X_k, the flow on path j is obviously $x_{kj}d_k$. This induces a flow on link ℓ of $x_{kj}d_k$ if ℓ is on path j and zero otherwise. Summing over all paths and all OD pairs, we have:

Flow on link ℓ induced by $X = (X_1 \ldots, X_p)$:

$$f_\ell(X) = \sum_k \left(\sum_j a_{\ell j}^{(k)} x_{kj} \right) d_k .$$

The (average) cost (or time) to traverse link ℓ in the network is assumed to vary with the flow on the link, and to be nondecreasing with the amount of the flow.

Average cost for flow f_ℓ on link ℓ: $c_\ell(f_\ell)$, a continuous non-decreasing function defined for $f_\ell \geqq 0$.

The (average) costs on the links are assumed to be additive on paths. Intuitively, the time to traverse a path is the sum of the times needed to traverse its links.

Average cost of path j joining OD pair k induced by proportionate flows $X = (X_1, \ldots, X_p)$:

$$C_{kj}(X) = \sum_\ell a_{\ell j}^{(k)} c_\ell(f_\ell(x)) = \sum_\ell a_{\ell j}^{(k)} c_\ell\left(\sum_k \left(\sum_j a_{\ell j}^{(k)} x_{kj}\right) d_k\right).$$

The reader who wishes to test his mastery of this notation may find it helpful to turn to the example of Section 7 at this point.

Two equilibrium concepts have been used in traffic analysis since they were introduced by Wardrop [6] in the early 50's. The first is descriptive and is used when it is assumed that each traveler adjusts so as to minimize his own travel time; consequently, it is called the User Equilibrium. Formally:

USER EQUILIBRIUM: $\bar{X} = (\bar{X}_1, \ldots, \bar{X}_p)$ such that if $\bar{x}_{kh} > 0$ then $C_{kh}(\bar{X}) = \min_j C_{kj}(\bar{X})$.

The second equilibrium concept is normative and uses a total system cost

$$C(X) = \sum_\ell c_\ell(f_\ell(X))\, f_\ell(X);$$

it is called the System Equilibrium. Formally:

SYSTEM EQUILIBRIUM: $\hat{X} = (\hat{X}_1, \ldots, \hat{X}_p)$ such that

$$C(\hat{X}) = \min_X C(X).$$

For some time, it has been recognized that the conditions for a User Equilibrium are merely the Kuhn-Tucker conditions for an associated nonlinear program. Since this program asks for the minimum of a convex function subject to linear constraints, the conditions are both necessary and sufficient. Precisely, $\phi_\ell(f_\ell) = \int_0^{f_\ell} c_\ell(f)df$. This is a convex function and is strictly convex where c_ℓ is strictly increasing. Define $\Phi(X) = \sum_\ell \phi_\ell(f_\ell(X))$. It is then a trivial exercise to prove:

THEOREM 2.1: $\bar{X}$ is a User Equilibrium if and only if $\Phi(\bar{X}) = \min_X \Phi(X)$.

This theorem lies at the base of attempts to find User Equilibria by nonlinear programming algorithms (such as the Frank-Wolfe algorithm for convex programming). In contrast, we shall use the definition of a User Equilibrium directly.

3. Labelling Proportionate Flows

In this section we shall define an appropriate labelling of points in space of proportionate flows: $S_{n_1} \times \cdots \times S_{n_p}$. The dimension of this space is $n_1 + \ldots + n_p - p$ and hence, by analogy with the example of Section 1,

we would expect the labels to run from 1 to $N = n_1 + \cdots + n_p - p + 1$, since a simplex of dimension $N - 1$ has N vertices. However, to make a direct connection with the definition of a User Equilibrium, we shall introduce initially labels which are vectors $(j_1, \ldots, j_p)$ where $j_k \in \{1, \ldots, n_k\}$ and $k = 1, \ldots, p$. There are, of course, $n_1 n_2 \ldots n_p$ such labels.

VECTOR LABEL RULE: The <u>vector label</u> of a flow $X = (X_1, \ldots, X_p)$ is denoted by

$$L(X) = (j_1, \ldots, j_p)$$

where $C_{kj_k}(X) = \max_{j, x_{kj} > 0} C_{kj}(X)$.

for $k = 1, \ldots, p$. If there are ties, use any definite rule to choose an eligible index, say the largest.

In words, the k^{th} component of $L(X)$ is the index of a path with positive flow that is most expensive among those with positive flow joining OD pair k.

COMPLETENESS FOR VECTOR LABELS: A set $\mathcal{L}$ of vector labels is said to be <u>complete</u> if, for all k and $j \in \{1, \ldots, n_k\}$, there exists $L = (j_1, \ldots, j_p) \in \mathcal{L}$ such that $j_k = j$.

We now define a transformation of vector labels into integer labels chosen from $\{1, \ldots, N\}$.

CONVERSION OF VECTOR LABELS TO INTEGER LABELS: Given $L = (j_1, \ldots, j_p)$ where $j_k \in \{1, \ldots, n_k\}$ for $k = 1, \ldots, p$, the associated <u>integer label</u> IL is defined by:

$$IL = \begin{cases} n_1 + \cdots + n_k + j_{k+1} - k & \text{where } k \text{ is the smallest index, } 0 \leq k \leq p-1, \text{ for which } j_{k+1} < n_{k+1}. \\ n_1 + \cdots + n_p + 1 - p & \text{if } j_k = n_k \text{ for } k = 1, \ldots, p. \end{cases}$$

We shall now establish some properties of this conversion of vector labels to integer labels.

LEMMA 3.1: If the vector labels L are ordered lexicographically and the integer labels IL are given their natural order $1, 2, \ldots, N$ then the transformation $I: L \rightarrow IL$ is monotonic nondecreasing.

PROOF. Suppose $L = (j_1, \ldots, j_p)$ precedes $L' = (j_1', \ldots, j_p')$ in the lexicographic order, written

$$L \lesssim L'.$$

If $j_1' = n_1, \ldots, j_p' = n_p$ then $IL' = N$ and $IL \leqq IL'$. Otherwise, let k' be the smallest index for which $j'_{k'+1} < n_{k'+1}$ and let k be the smallest index for which $j_{k+1} < n_{k+1}$. Then

$$IL' = n_1 + \cdots + n_{k'} + j'_{k'+1} - k'$$

$$IL = n_1 + \cdots + n_k + j_{k+1} - k.$$

If $k = k'$ then $j_{k+1} \leqq j'_{k+1}$ since $L \lesssim L'$ and hence $IL \leqq IL'$. If $k < k'$ then

$$IL' - IL = n_{k+1} + \cdots + n_{k'} + j'_{k'+1} - j_{k+1} - (k' - k)$$

$$= (n_{k+1} - j_{k+1}) + (n_{k+2} + \cdots + n_{k'}) + j'_{k'+1} - (k' - k)$$

Note $n_{k+1} - j_{k+1} \geqq 1$, $n_{k+2} + \cdots + n_{k'} \geqq k' - (k+1)$, $j'_{k'+1} \geqq 1$

and hence $IL' - IL \geqq 1 + (k' - k - 1) + 1 - (k' - k) = 1$.

QED

LEMMA 3.2: For all $j \in \{1, \ldots, N-1\}$ there exist unique $k \in \{0, \ldots, p-1\}$ and $j_{k+1} \in \{1, \ldots, n_{k+1}\}$ such that $j_{k+1} < n_{k+1}$ and $j = n_1 + \ldots + n_k + j_{k+1} - k$.

PROOF. The existence is proved by induction on j.

For $j = 1 = n_1 + \ldots + n_k + j_{k+1} - k$

$$\geq (n_1 - 1) + \cdots + (n_k - 1) + j_{k+1} \geq k + j_{k+1}$$

implies the unique solution $k = 0$ and $j_{k+1} = 1$.

Suppose $j < N - 1$ has been expressed as

$$j = n_1 + \cdots + n_k + j_{k+1} - k.$$

If $j_{k+1} < n_{k+1} - 1$ then

$$j + 1 = n_1 + \cdots + n_k + (j_{k+1} + 1) - k$$

provides the required solution. If $j_{k+1} = n_{k+1} - 1$ then

$$j + 1 = n_1 + \cdots + n_{k+1} + 1 - (k+1)$$

provides the required solution.

To prove uniqueness, suppose two solutions

$$j = n_1 + \cdots + n_k + j_{k+1} - k$$

$$j = n_1 + \cdots + n_{k'} + j'_{k'+1} - k'$$

where we may assume $k \leq k'$ without loss of generality. If $k = k'$ then $j_{k+1} = j'_{k'+1}$. Otherwise $k < k'$ and

$$0 = (n_{k+1} - j_{k+1}) + (n_{k+2} + \cdots + n_{k'}) + j'_{k'+1} - (k' - k)$$

Note $n_{k+1} - j_{k+1} \geq 1$, $n_{k+2} + \cdots + n_{k'} \geq k' - (k+1)$, $j'_{k'+1} \geq 1$ and hence $0 \geq 1 + k' - (k+1) + 1 - (k' - k) = 1$, a contradiction.

QED

COROLLARY. If $IL = j \in \{1, \ldots, N-1\}$ then $L = (n_1, \ldots, n_k, j_{k+1}, *, \ldots, *)$ where k and j_{k+1} are unique as in Lemma 3.2. If $IL = N$ then $L = (n_1, \ldots, n_p)$.

THEOREM 3.1: Suppose $\mathcal{L}$ is a set of vector labels such that $I\mathcal{L} = \{1, \ldots, N\}$. Then $\mathcal{L}$ is complete.

PROOF. This follows directly from the Corollary. Given any k and $j \in \{1, \ldots, n_k\}$ we have three cases:

Case 1. $j < n_k$ Then $n_1 + \ldots + n_{k-1} + j - (k-1) \in I\mathcal{L}$ and $(n_1, \ldots, n_{k-1}, j, *, \ldots, *) \in \mathcal{L}$.

Case 2. $j = n_k$, $k < p$ Then $n_1 + \ldots + n_k + 1 - k \in I\mathcal{L}$ and $(n_1, \ldots, n_k, 1, *, \ldots, *) \in \mathcal{L}$.

Case 3. $j = n_p$, $k = p$ Then $n_1 + \ldots + n_p + 1 - p \in I\mathcal{L}$ and $(n_1, \ldots, n_p) \in \mathcal{L}$.

QED.

Theorem 3.1 asserts that completeness in induced integer labels implies completeness of the vector labels. The converse is not true as the following example shows:

EXAMPLE 3.1: $n_1 = n_2 = 2$. Let $\mathcal{L} = \{(1,2), (2,1)\}$: clearly $\mathcal{L}$ is complete. However, $I\mathcal{L} = \{1, 2\}$ is not complete since it lacks $N = 3$.

As we shall see in the following sections, Theorem 3.1 plays a crucial role in the algorithm. However, the transformation I: $L \rightarrow IL$ has a property that affects the performance of the algorithm adversely, namely, there is only one vector label that receives the induced integer label N, namely, $(n_1, \ldots, n_p)$. On the other hand there are many vector labels

(namely, all vector labels of the form $(1, *, \ldots, *)$) that induce the integer label 1. We would prefer a transformation that is more "balanced" while preserving the essential result of Theorem 3.1 that completeness in induced integer labels implies completeness in the vector labels from which they come. This seems to be a difficult combinatorial problem.

4. Labelling and User Equilibrium

In Section 3, we have given a specific rule for assigning a vector label to a proportionate flow. This was done without motivation or explanation. In this section, the vital connection between labelling and User Equilibrium will be established. This will be done in two parts, first establishing a necessary property of User Equilibria, then showing that this property, expressed in terms of our labelling rule is sufficient for a User Equilibrium.

THEOREM 4.1: Let $\bar{X}$ be a User Equilibrium. Then, for all k, all $h \in \{1, \ldots, n_k\}$, and all $\epsilon > 0$, there exists X (depending on k, h, and ϵ) such that $X \in S_{n_1} \times \cdots \times S_{n_p}$, $|X - \bar{X}| < \epsilon$, $C_{kh}(X) = \max_{j, x_{kj} > 0} C_{kj}(X)$, and $x_{kh} > 0$.

PROOF. Informally, this theorem says that in every ϵ-neighborhood of a User Equilibrium there exists a flow that makes any specified path joining any specified OD pair have a positive flow and at the same time be the most expensive among the paths with positive flow joining that OD pair. This is obviously true for those paths with $\bar{x}_{kh} > 0$ since for all of these paths,

$$C_{kh}(\bar{X}) = \min_{j, \bar{x}_{kj} > 0} C_{kj}(\bar{X}),$$ a constant, and the point $\bar{X}$ itself satisfies the assertion of the theorem.

Suppose $\bar{x}_{kh} = 0$ and $C_{kh}(\bar{X}) > \min_{j, \bar{x}_{kj} > 0} C_{kj}(\bar{X})$.

We may assume, without loss of generality, that $\bar{x}_{kl} > 0$. Then $X = \left(\bar{X}_1, \ldots, \left(\bar{x}_{kl} - \frac{\epsilon}{2}, \ldots, \bar{x}_{kh} + \frac{\epsilon}{2}, \ldots\right), \ldots, \bar{X}_p\right)$ satisfies the assertion of the theorem.

This leaves the case $\bar{x}_{kh} = 0$, $C_{kh}(\bar{X}) = \min_{j, \bar{x}_{kj} > 0} C_{kj}(\bar{X})$, that is, of a path that is not in use even though its cost is a minimum among the paths joining OD pair k. If we increase the flow on this path by $\epsilon/2$, while decreasing the flow by $\epsilon/2$ on some other path with positive flow in $\bar{X}$ (say $\bar{x}_{kl}$), as in the preceding case, then $C_{kh}(X) \geq C_{kj}(X)$ for all j with $\bar{x}_{kj} > 0$. This follows from the fact that the $c_\ell(f_\ell)$ are increasing functions. The flow has been increased by $\epsilon/2$ on those links in path h and not in path l joining OD pair k and decreased by $\epsilon/2$ on those links in path l and not in path h. Path h contains all of the links for which the flow has been increased, and none of the links on which the flow has been decreased. (Other paths with positive flow may have the same property, which accounts for the possibility that for such a path j, $C_{kh}(X)$ may equal $C_{kj}(X)$.) However path h now definitely has the maximum cost of those paths with positive flow joining OD pair k.

QED

Of course, it is the converse to this theorem that motivates the labelling rule given in Section 3. This is:

THEOREM 4.2: Suppose $\bar{X} \in S_{n_1} x \ldots x S_{n_p}$ has the following property: For all k, all $h \in \{1, \ldots, n_k\}$, and all $\epsilon > 0$, there exists X (depending on k, h, and ϵ) such that $X \in S_{n_1} x \ldots x S_{n_p}$, $|X - \bar{X}| < \epsilon$, and $C_{kh}(X) = \max_{j, x_{kj} > 0} C_{kj}(X)$. Then $\bar{X}$ is a User Equilibrium.

PROOF. Suppose $\bar{x}_{kj} > 0$ and $\bar{x}_{kj'} > 0$. Then in every ϵ-neighborhood of $\bar{X}$, for $\epsilon > 0$ sufficiently small, $x_{kj} > 0$ and $x_{kj'} > 0$. Hence we may conclude $C_{kj}(\bar{X}) \geq C_{kj'}(\bar{X})$ and $C_{kj'}(\bar{X}) \geq C_{kj}(\bar{X})$, that is, $C_{kj}(\bar{X})$ is a constant, $\bar{C}$, for all paths j with $\bar{x}_{kj} > 0$.

Suppose there is a path h with $\bar{x}_{kh} = 0$ and $C_{kh}(\bar{X}) < \bar{C}$. Then, for $\epsilon > 0$ sufficiently small, and $|X - \bar{X}| < \epsilon$, $C_{kh}(X) < C_{kj}(X)$ for all j with $\bar{x}_{kj} > 0$, by the continuity of the functions C_{kj}. However, this contradicts the hypothesis of the theorem. Hence $C_{kh}(\bar{X}) \geq \bar{C}$ for all paths h with $\bar{x}_{kh} = 0$.

QED

Finally, we establish the connection with the labelling rule given in Section 3.

THEOREM 4.3: Suppose $\bar{X} \epsilon S_{n_1} \times \ldots \times S_{n_p}$ has the following property: In every ϵ-neighborhood of $\bar{X}$ there exist flows with a complete set of integer labels $1, 2, \ldots, n_1 + \ldots + n_p + 1 - p$. Then $\bar{X}$ is a User Equilibrium.

PROOF. This theorem is an immediate consequence of Theorems 3.1 and 4.2. Theorem 3.1 says that completeness in integer labels implies completeness in vector labels. Theorem 4.2 says that completeness in vector labels in every ϵ-neighborhood of $\bar{X}$ implies that $\bar{X}$ is a User Equilibrium.

QED

5. Subdivision and Restarting

As was noted at the beginning of Section 3, the dimension of the space $S_{n_1} \times \ldots \times S_{n_p}$ of proportionate flows is $n_1 + \ldots + n_p - p$. Theorem 4.3 suggests that, to find User Equilibria, we should seek sets of $n_1 + \ldots + n_p + 1 - p$ points, very close together, with all integer labels. Following the general techniques of "label and pivot" methods, we need a subdivision of $S_{n_1} \times \ldots \times S_{n_p}$ into simplices of dimension $n_1 + \ldots + n_p - p$, hence with the appropriate number $(n_1 + \ldots + n_p + 1 - p)$ of vertices. The subdivision given here has been designed for efficient computer implementation.

SUBDIVISION OF $S_{n_1} \times \ldots \times S_{n_p}$ OF DEGREE $(D_1, \ldots, D_p)$:

Let D_k be a positive integer, $k = 1, \ldots, p$. A vertex of the subdivision is a vector of nonnegative integers z_{kj}

$$Z = (Z_1, \ldots, Z_p) = \left((z_{11}, \ldots, z_{1n_1}), \ldots, (z_{p1}, \ldots, z_{pn_p})\right)$$

where $\sum_j z_{kj} = D_k$ for $k = 1, \ldots, p$. (Note that $x_{kj} = z_{kj}/D_k$ defines the corresponding proportionate flow; for computer work it is easier to work in integers.) A simplex of the subdivision is a set of N vertices, where $N = n_1 + \ldots + n_p + 1 - p$, that can be arranged as:

$$Z^{(1)} = \left(Z_1^{(1)}, \ldots, Z_p^{(1)}\right)$$

$$\ldots \qquad \ldots \qquad \ldots$$

$$Z^{(N)} = \left(Z_1^{(N)}, \ldots, Z_p^{(N)}\right)$$

with

$$Z^{(n+1)} = Z^{(n)} + (\ldots, E_{k,j+1} - E_{k,j}, \ldots)$$

for $n = 1, \ldots, N - 1$. In this definition, $E_{k,j}$ is the j^{th} unit vector of S_{n_k}. Every pair (k,j) is to appear in this sequence exactly once for $k = 1, \ldots, p$ and $j = 1, \ldots, n_k - 1$.

The application of this definition to the p-cube, $S_2 x \ldots x S_2$ (the product of p-unit intervals), coincides with the subdivision given in [3]. The manner in which it generalizes this subdivision is illustrated in the following example:

EXAMPLE 5.1: The subdivision of $S_2 \times S_3$ of degree (1, 1). The three simplices of the subdivision are:

	Simplex 1	Simplex 2	Simplex 3
$Z^{(1)}$:	$\big((1,0), (1,0,0)\big)$	$\big((1,0), (1,0,0)\big)$	$\big((1,0), (1,0,0)\big)$
$Z^{(2)}$:	$\big((0,1), (1,0,0)\big)$	$\big((1,0), (0,1,0)\big)$	$\big((1,0), (0,1,0)\big)$
$Z^{(3)}$:	$\big((0,1), (0,1,0)\big)$	$\big((0,1), (0,1,0)\big)$	$\big((1,0), (0,0,1)\big)$
$Z^{(4)}$:	$\big((0,1), (0,0,1)\big)$	$\big((0,1), (0,0,1)\big)$	$\big((0,1), (0,0,1)\big)$

This subdivision is shown in Figure 5.1 with the three simplices extracted below it.

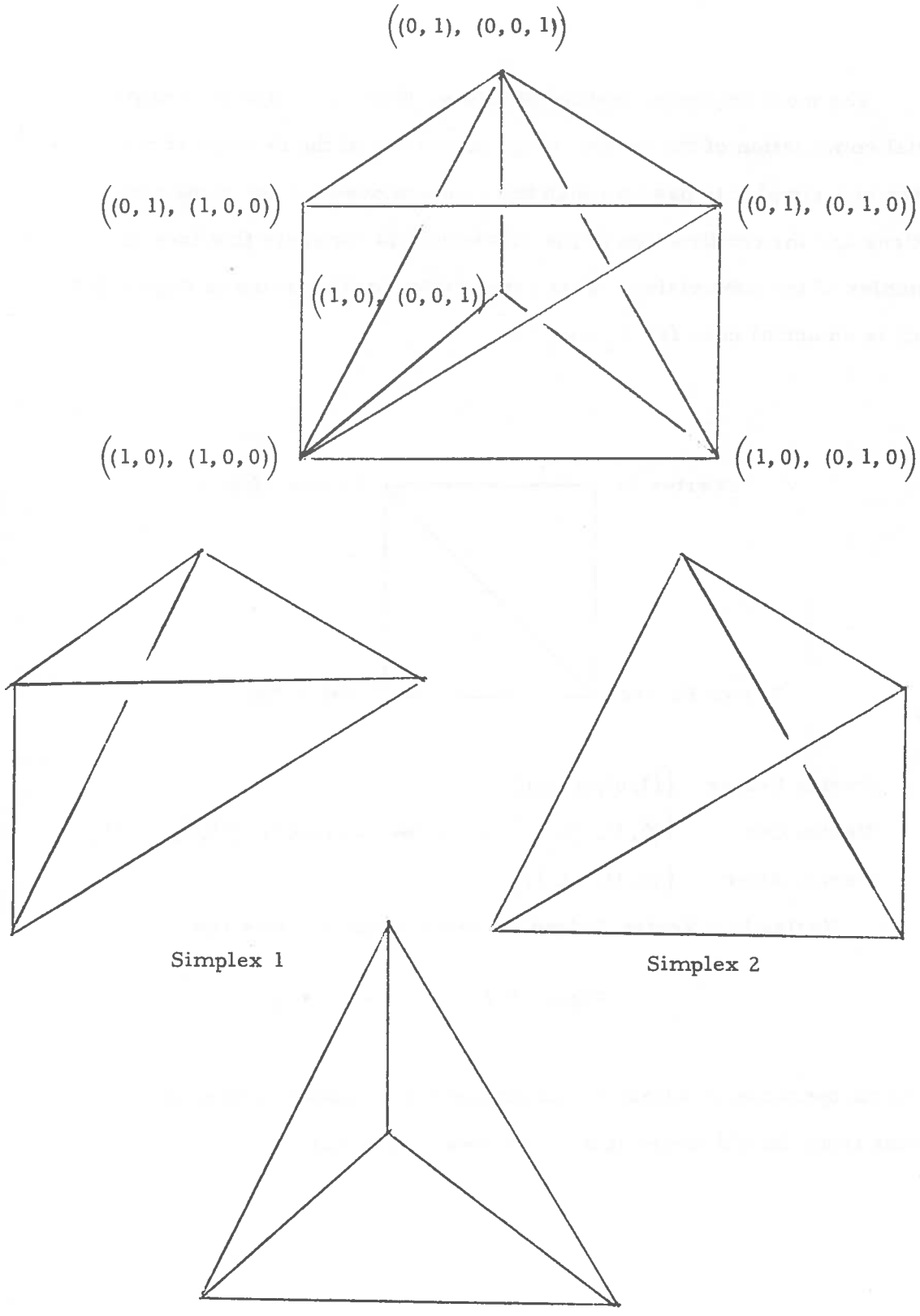
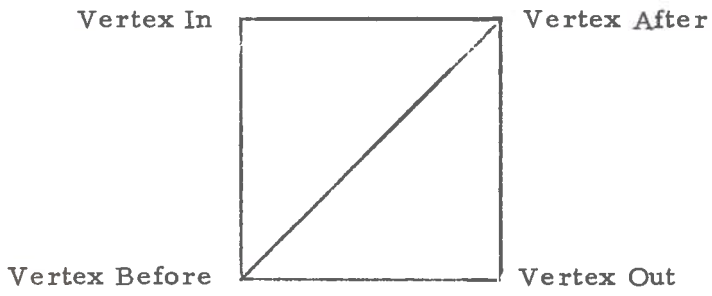

((0, 1), (0, 0, 1))
((0, 1), (1, 0, 0))
((0, 1), (0, 1, 0))
((1, 0), (0, 0, 1))
((1, 0), (1, 0, 0))
((1, 0), (0, 1, 0))
Simplex 1
Simplex 2
Simplex 3

Figure 5.1

The most important feature of this subdivision is that it permits a trivial computation of the "pivot" step, consisting of the deletion of one vertex in a simplex to pass through the face composed of the remaining vertices and the construction of the new vertex to complete this face to a simplex of the subdivision. This computation is illustrated in Figure 5.2, which is an actual case for $n_1 = n_2 = 2$.



Vertex Before $\big((1,0),\ (1,0)\big)$

Vertex Out $\big((0,1),\ (1,0)\big)$ ⟶ Vertex In $\big((1,0),\ 0,1)\big)$

Vertex After $\big((0,1),\ (0,1)\big)$

Vertex In = Vertex Before + Vertex After - Vertex Out

Figure 5.2

The same operation is illustrated in Example 5.1 above by arrows pointing from the old vertex (out) to the new vertex (in).

PIVOT OPERATION: Given a simplex $Z^{(1)}, \ldots, Z^{(N)}$ of the subdivision, and a vertex $Z^{(n)}$ of this simplex, if the remaining vertices do not lie on the boundary of $S_{n_1} x \ldots x S_{n_p}$ (that is, if no fixed component of the remaining vertices equals 0 throughout) then there is a unique simplex of the subdivision that shares the remaining vertices. It is composed of these vertices and a new vertex

$$\text{(Vertex In) } Z = Z^{(n-1)} + Z^{(n+1)} - Z^{(n)} \text{ (Vertex Out).}$$

(In this definition, the conventions $Z^{(0)} = Z^{(N)}$ and $Z^{(N+1)} = Z^{(1)}$ are used to cover the extreme cases of $n = 1$ and $n = N$.)

The proof that the subdivision given is a valid simplicial subdivision will not be given here since it parallels in some detail arguments already presented in [3]. In essence, one needs to show that every point of $S_{n_1} x \ldots x S_{n_p}$ belongs to at least one of the simplices and that the pivot operation, as defined is unique.

It is easy to give the number of simplices in a subdivision of degree $(D_1, \ldots, D_p)$. The product subdivision gives $D_1^{n_1-1} D_2^{n_2-1} \ldots D_p^{n_p-1}$ product sets of degree $(1, \ldots, 1)$ and each is cut into $\frac{(n_1+\ldots+n_p-p)!}{(n_1-1)! \ldots (n_p-1)!}$ pieces. For example, a subdivision of $S_2 \times S_2 \times S_2$ of degree 10 has 6,000 tetrahedra. Fortunately, in practice, the algorithm only examines a small fraction of the pieces.

As was remarked at the beginning of this section, we can approximate a User Equilibrium if we can find sets of $N = n_1+\ldots+n_p+1-p$ points arbitrarily close together with a complete set of integer labels. The existence of such sets is asserted by the next theorem. It will not be proved at this point; instead, a constructive argument that parallels the algorithm is given following the introduction of the "restart" concept.

THEOREM 5.1: Given a subdivision of $S_{n_1} x \ldots x S_{n_p}$ of degree $(D_1, \ldots, D_p)$ and a (vector) labelling of the vertices such that $L(X_1, \ldots, X_p) = (j_1, \ldots, j_p)$ implies $x_{kj_k} > 0$ for $k = 1, \ldots, p$, there exists at least one simplex of the subdivision with a complete set of induced integer labels.

Although we shall not prove this theorem at this point, it is appropriate to draw a corollary. If we observe that two vertices of a simplex of the subdivision (in coordinates z_{kj}) cannot differ by more than 1 in any coordinate, then the associated flows cannot differ by more than $1/D_k$ in any coordinate. Hence, two flows that are vertices of a simplex in $S_{n_1} x \ldots x S_{n_k}$ cannot be further than

$$\sum_k \sqrt{n_k / D_k}$$

apart. Hence, if we choose $D_1^t, \ldots, D_p^t$ such that $\lim_{t \to \infty} D_k^t = \infty$ for $k = 1, \ldots, p$, the diameter of each piece tends to zero. Let $\bar{X}^t$ be the center of gravity of a completely labelled simplex, whose existence is asserted in Theorem 5.1. As a sequence of points in the compact set, there exists a subsequence $\bar{X}^{t_s}$ that converges to a point $\bar{X}$. In every neighborhood of $\bar{X}$ there exist flows with a complete set of integer labels. Hence $\bar{X}$ is a User Equilibrium. With slightly more attention to detail it is possible to prove the following result; the proof will not be given here since it parallels the proof of Proposition 2.3 of [5] quite closely.

THEOREM 5.2: Given $\epsilon > 0$, there exists D_o such that for $D_k \geq D_o$ for $k = 1, \ldots, p$ and X^* any proportionate flow in a completely labelled simplex of the subdivision of degree $(D_1, \ldots, D_p)$, then there exists a User Equilibrium $\bar{X}$ such that $|X^* - \bar{X}| < \epsilon$.

COROLLARY. Suppose the $c_\ell(f_\ell)$ are strictly increasing. Then, under the conditions of Theorem 5.2, X^* converges to the unique User Equilibrium as $D_o \to \infty$.

PROOF. If the $c_\ell(f_\ell)$ are strictly increasing then the nonlinear program equivalent to User Equilibrium (see Theorem 2.1) has a strictly convex objective function and hence a unique solution.

QED

We shall now present an infallible rule for starting the search for a completely labelled simplex in the subdivision of $S_{n_1} x \ldots x S_{n_p}$. This rule uses the idea of "homotopy" introduced by Eaves in 1971 [7], to connect to the problem to an easily solvable problem by what may be considered a piecewise linear connection. When used repeatedly, as is intended here, starting each iteration near the previously computed approximation, it is called "sandwiching" or "restarting," a method first used by Merrill in 1971 (see [8], [9]) and rediscovered independently by MacKinnon in 1972 (see [5]). We shall refer to [5] for the motivation of this method.

We shall now work on a product set of one extra dimension. Consider a subdivision of degree $(D_1, \ldots, D_p, 1)$ of $S_{n_1} x \ldots x S_{n_p} x S_2$. [Geometrically, this set consists of two duplicates of the space of proportionate flows, joined by an interval from (1, 0) to (0, 1).] The integer labelling of vertices of the form $\left(Z_1, \ldots, Z_p, (0,1)\right)$ is the same as in Section 3 and depends on the costs of flows specified by $(Z_1, \ldots, Z_p)$. The integer labelling of vertices of the form $\left(Z_1, \ldots, Z_p, (1,0)\right)$ is, in a sense, an "artificial" labelling designed to start the search from a preassigned location. Let $(\bar{Z}_1, \ldots, \bar{Z}_p)$ be

such that $\left(\bar{Z}_1 + E_{n_1}, \ldots, \bar{Z}_p + E_{n_p}\right)$ is the best available non-negative integer approximation to the equilibrium of degree $(D_1, \ldots, D_p)$. Note that $\Sigma_j \bar{Z}_{kj} = D_k - 1$ for $k = 1, \ldots, p$.

ARTIFICIAL VECTOR LABEL RULE:

$$L\left(Z_1, \ldots, Z_p, (1,0)\right) = \left(j_1, \ldots, j_p\right)$$

where j_k is the first maximum of $z_{kj} - \bar{z}_{kj}$.

Note that since $\Sigma_j z_{kj} = D_k$ and $\Sigma_j \bar{z}_{kj} = D_k - 1$, this maximum must be at least one. The vector labelling induces an integer labelling as before.

THEOREM 5.3: The unique simplex in the set $S_{n_1} x \ldots x S_{n_p} x \{(1,0)\}$ with a complete set of integer labels (induced by the artificial vector labelling) is listed below (with the integer labels at left):

IL	
1	$((\bar{z}_{11}+1, \bar{z}_{12}, \ldots, \bar{z}_{1n_1}), (\bar{z}_{21}+1, \bar{z}_{22}, \ldots, \bar{z}_{2n_2}), \ldots, (\bar{z}_{p1}+1, \bar{z}_{p2}, \ldots \bar{z}_{pn_p}), (1,0))$
2	$((\bar{z}_{11}, \bar{z}_{12}+1, \ldots, \bar{z}_{1n_1}), \bar{z}_{21}+1, \bar{z}_{22}, \ldots, \bar{z}_{2n_2}), \ldots, (\bar{z}_{p1}+1, \bar{z}_{p2}, \ldots, \bar{z}_{pn_p}), (1,0))$
	.
	.
	.
n_1	$((\bar{z}_{11}, \bar{z}_{12}, \ldots, \bar{z}_{1n_1}+1), (\bar{z}_{21}+1, \bar{z}_{22}, \ldots, \bar{z}_{2n_2}), \ldots, (\bar{z}_{p1}+1, \bar{z}_{p2}, \ldots, \bar{z}_{pn_p}), (1,0))$
n_1+1	$((\bar{z}_{11}, \bar{z}_{12}, \ldots, \bar{z}_{1n_1}+1), (\bar{z}_{21}, \bar{z}_{22}+1, \ldots, \bar{z}_{2n_2}), \ldots, (\bar{z}_{p1}+1, \bar{z}_{p2}, \ldots, \bar{z}_{pn_p}), (1,0))$
	.
	.
	.
$n_1+\ldots+n_p-p+1$	$((\bar{z}_{11}, \bar{z}_{12}, \ldots, \bar{z}_{1n_1}+1), (\bar{z}_{21}, \bar{z}_{22}, \ldots, \bar{z}_{2n_2}+1), \ldots, (\bar{z}_{p1}, \bar{z}_{p2}, \ldots, \bar{z}_{pn_p}+1), (1,0))$

This is completed to a simplex in the subdivision of $S_{n_1} x \ldots x S_{n_p} x S_2$ by the vertex

$$((\bar{z}_{11}, \bar{z}_{12}, \ldots, \bar{z}_{1n_1}+1), (\bar{z}_{21}, \bar{z}_{22}, \ldots, \bar{z}_{2n_2}+1), \ldots, (\bar{z}_{p1}, \bar{z}_{p2}, \ldots, \bar{z}_{pn_p}+1), (0,1)).$$

PROOF. It is clear that the given set of vectors forms a simplex in $S_{n_1} x \ldots x S_{n_p} x \{(1,0)\}$ and that the extra vector completes it to a simplex in $S_{n_1} x \ldots x S_{n_p} x S_2$. Furthermore, the integer labels given are easily computed from the rule for artificial vector labels and the transformation into integer labels. The argument that establishes uniqueness parallels the proof of Proposition 3.4 of [5] and will not be repeated here.

QED

As is customary in pivotal algorithms, the new vertex is labelled (with an integer from 1, ... , N), the unique vertex with a repeated label is dropped, and a new vertex enters by the pivotal rules given previously. In the present situation, the simplices are of dimension N and have N + 1 vertices, while the only labels that appear run from 1, ... , N. Thus the simplices that are generated have exactly two faces with a complete set of labels 1, ... , N. The algorithm starts with the unique face in $S_{n_1} x \ldots x S_{n_p} x \{(1,0)\}$ with a complete set of labels. Hence, it cannot return to a face in this set. Furthermore, it cannot encounter a completely labelled face in the boundaries of $S_{n_1} x \ldots x S_{n_p} x S_2$ determined by $z_{kj} = 0$ for k = 1, ... , p, since these faces omit the integer label determined by Lemma 3.2. Hence, in a finite number of pivots the sequence must pass through the boundary of $S_{n_1} x \ldots x S_{n_p} x S_2$ through a

face in $S_{n_1} x \ldots x S_{n_p} x \{(0,1)\}$. This is signalled by the calculation of a new entering vertex of the form $(Z_1, \ldots, Z_p, (-1,2))$. When this occurs, the remaining vertices in $S_{n_1} x \ldots x S_{n_p} x \{(0,1)\}$ carry a complete set of integer labels. This completes a constructive proof of Theorem 5.1.

Of course, the algorithm is repeated at this point, using as the new approximation the center of gravity of the simplex just found and increasing the degree of the subdivision. The increase in degree is a parameter to be specified as is the initial degree. The algorithm may be terminated in a variety of ways. If all paths with non-negligible positive flow have nearly the same cost and this is significantly less than the cost for paths with a negligible flow, then we may consider that we have "nearly" an equilibrium. More customarily, the algorithm terminates with the degrees D_k are too large for the computer to handle. Finally, economic considerations may dictate a time limit.

6. A Flow Chart for the Algorithm

The flow chart shown in Figure 6.1 assembles the pieces of theory we have presented. Notes attached to the flow chart refer to previous sections as follows:

(A) Read in data. (The data consists of the following, described in Section 2: OD pairs, demand for OD pairs, links, link-path incidence matrices, link cost functions, plus initial degree of subdivision (Section 5), and an initial approximation (Section 5).)

(B) Determine the initial simplex and a pivot vector adjacent to the simplex. (This is described in Theorem 5.3.)

(C) Determine link flows for the pivot vector, then average link costs. These give average path costs. (The relevant formulas are given in Section 2.)

(D) For each OD pair determine the path with greatest average path cost and positive flow. If ties, take the path with the largest index. These indices yield a vector label. (This follows the vector labelling rule given in Section 3.)

(E) Determine the integer label corresponding to the vector label. (Follow the rule given for conversion from vector labels to integer labels given in Section 3.) The vector in the simplex which already has this integer label is pivoted off of to the next simplex. Determine the new vertex using the pivot rules. (The pivot rule is given in Section 5.)

(F) Artifical Vector Label: For each OD pair determine the largest positive gain in flow over initial approximation to problem. If ties, take the smallest such index of all such paths. (This follows the Artificial Vector Label Rule given in Section 5.)

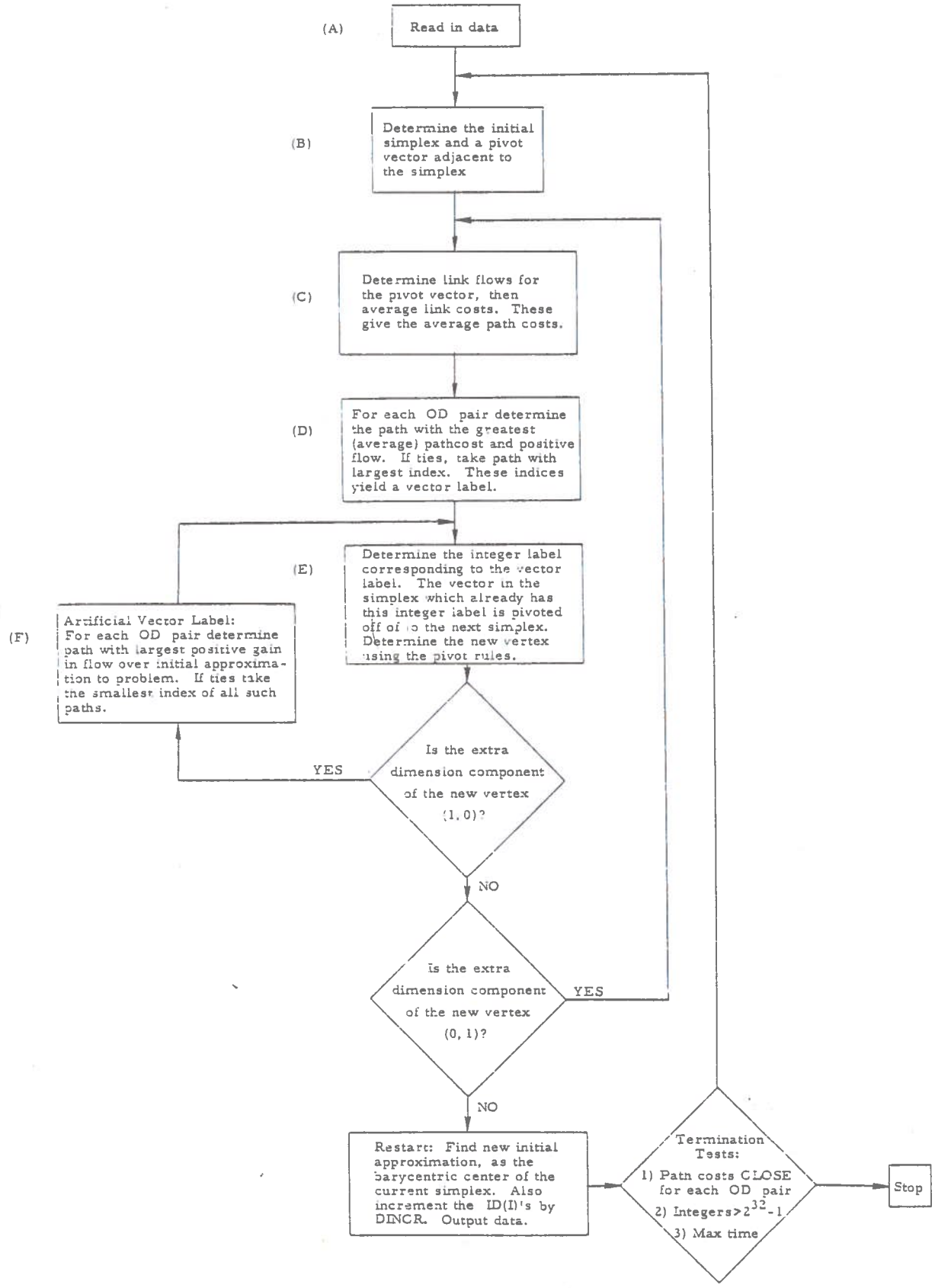
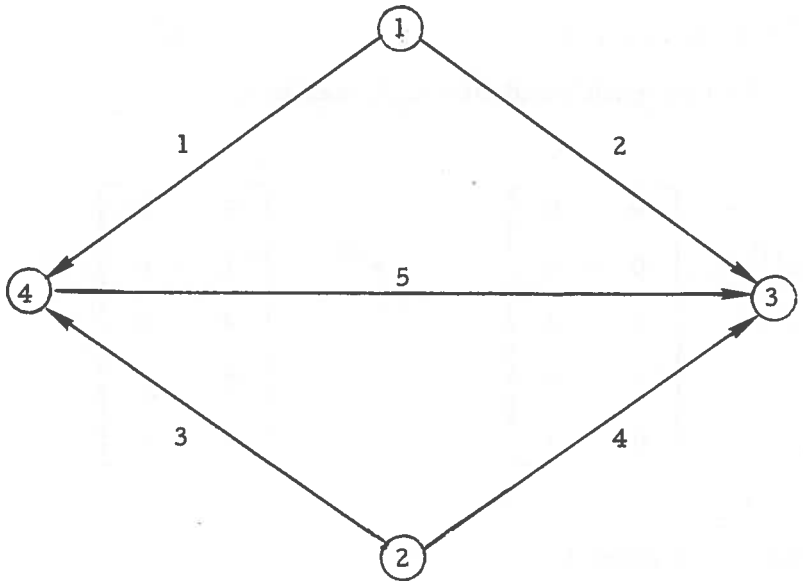

(A)
Read in data
(B)
Determine the initial simplex and a pivot vector adjacent to the simplex
(C)
Determine link flows for the pivot vector, then average link costs. These give the average path costs.
(D)
For each OD pair determine the path with the greatest (average) pathcost and positive flow. If ties, take path with largest index. These indices yield a vector label.
(E)
Determine the integer label corresponding to the vector label. The vector in the simplex which already has this integer label is pivoted off of to the next simplex. Determine the new vertex using the pivot rules.
(F)
Artificial Vector Label: For each OD pair determine path with largest positive gain in flow over initial approximation to problem. If ties take the smallest index of all such paths.
Is the extra dimension component of the new vertex (1, 0)?
YES
NO
Is the extra dimension component of the new vertex (0, 1)?
YES
NO
Restart: Find new initial approximation, as the barycentric center of the current simplex. Also increment the ID(I)'s by DINCR. Output data.
Termination Tests:
1) Path costs CLOSE for each OD pair
2) Integers>$2^{32}-1$
3) Max time
Stop

Figure 6.1

7. An Example

Consider the example with the following underlying network (note that there is no reason to index the nodes, although for the purposes of identifying OD pairs on the network we have done so).



In the notation of Section 2, we have the following data:

OD pairs: k = 1, 2. k = 1 corresponds to the node pair (2, 3) and k = 2 corresponds to the node pair (1, 3).

Demands: $d_1 = 2$, $d_2 = 2$.

Paths giving the OD pairs:

	Path	Links
OD pair 1	(1, 1)	4
	(1, 2)	3-5
OD pair 2	(2, 1)	2
	(2, 2)	1-5

Links: $\ell = 1, 2, \ldots, 5$

Therefore the link-path incidence matrices are:

$$A^{(1)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad A^{(2)} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

Proportionate flows are given by:

$$X = \left((x_{11}, x_{12}) , (x_{21}, x_{22}) \right)$$

where all $x_{kj} \geq 0$ and $x_{11} + x_{12} = 1$ and $x_{21} + x_{22} = 1$. Geometrically, these form the unit square or $S_2 \times S_2$. Flows on links induced by a proportionate flow are:

$$
\begin{aligned}
f_1 &= 2x_{22} \\
f_2 &= 2x_{21} \\
f_3 &= 2x_{12} \\
f_4 &= 2x_{11} \\
f_5 &= 2x_{12} + 2x_{22} \,.
\end{aligned}
$$

Average costs for a flow f_ℓ on link ℓ are given as quadratic functions $A_\ell + B_\ell f_\ell + C_\ell f_\ell^2$ where

Link	A_ℓ	B_ℓ	C_ℓ
1	2	0	0
2	15	0	0
3	4	0	0
4	16	0	0
5	0	1	1

That is, with the exception of link 5, average costs are constant, while the cost on link 5 is $f_5 + f_5^2$ where f_5 is the flow on link 5.

Average costs for the paths induced by a proportionate flow $X = \left((x_{11}, x_{12}), (x_{21}, x_{22})\right)$ are:

Path	Average Cost
(1, 1)	16
(1, 2)	$4 + 2(x_{12} + x_{22}) + 4(x_{12} + x_{22})^2$
(2, 1)	15
(2, 2)	$2 + 2(x_{12} + x_{22}) + 4(x_{12} + x_{22})^2$

This completes the application of the notation introduced in Section 2. To analyze the labelling with a simplification of notation, let $x = x_{11}$, $x_{12} = 1 - x$, $y = x_{21}$, $x_{22} = 1 - y$. Then the average costs of paths become:

$$C_{11}(X) = 16$$
$$C_{12}(X) = 4 + 2(2 - x - y) + 4(2 - x - y)^2$$
$$C_{21}(X) = 15$$
$$C_{22}(X) = 2 + 2(2 - x - y) + 4(2 - x - y)^2$$

It is easily verified that $C_{11} = C_{12}$ when $x + y = 0.50$ and $C_{21} = C_{22}$ when $x + y = 0.43$. Therefore, the vector labelling rule of Section 3 produces Figure 7.1.

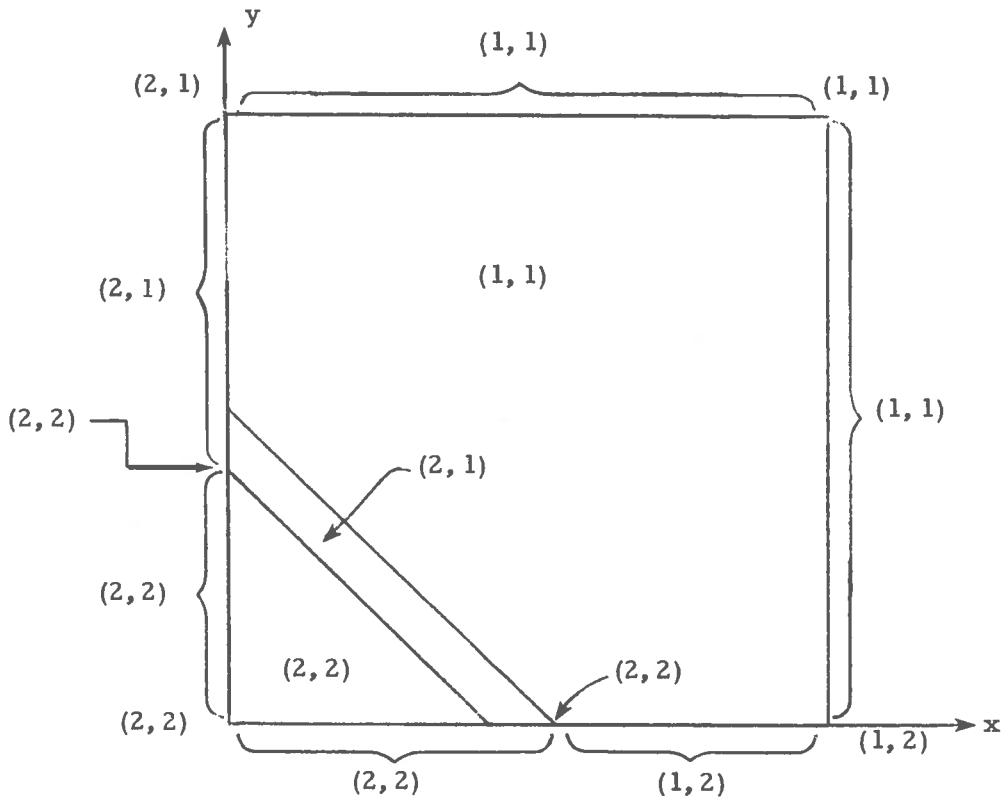


Figure 7.1

Note that ties always go to the higher index and so the segment $x + y = 0.50$ is labelled $(2, 1)$ except when $x = 0.50$, $y = 0$ where it is labelled $(2, 2)$. The rule for converting vector labels to integer labels for this example is summarized by the table:

L		IL
$(1, 1)$	$\longrightarrow$	1
$(1, 2)$	$\longrightarrow$	1
$(2, 1)$	$\longrightarrow$	2
$(2, 2)$	$\longrightarrow$	3

Hence, the square labelled by integers is as shown in Figure 7.2.

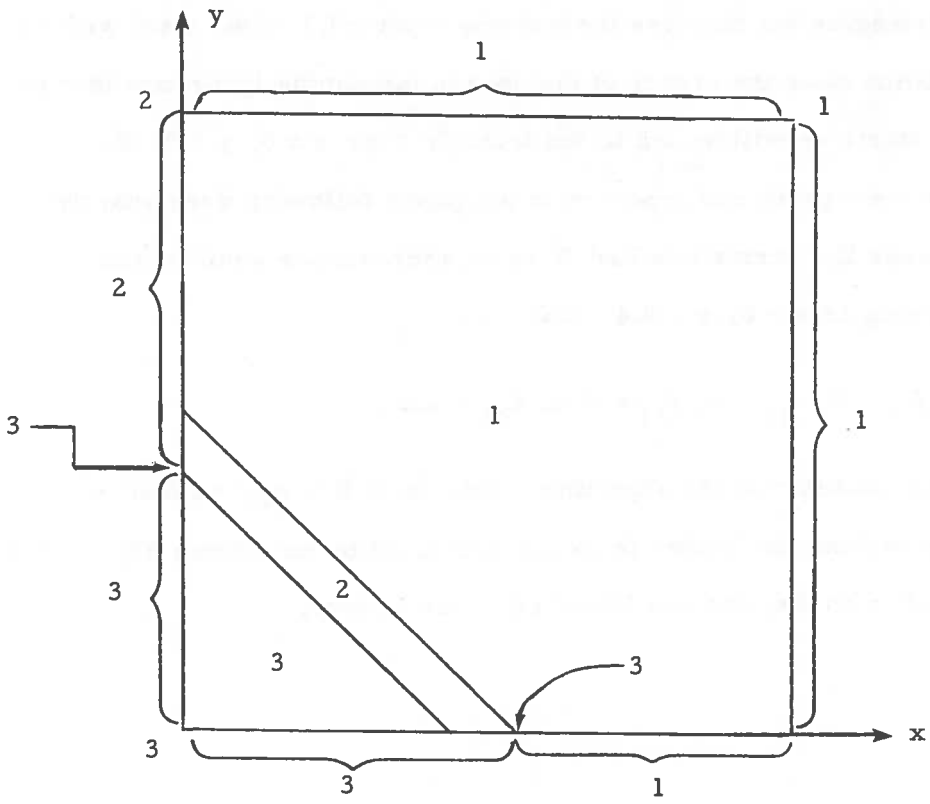


Figure 7.2

Note that again, ties go to the the higher index on boundaries between two open sets (of any dimension) with different indices.

Figure 7.2 illustrates vividly the content of Theorem 4.3. Clearly the point $x = 0.50$, $y = 0$ is the only point in the square which has all labels in every ϵ-neighborhood and, therefore, is the User Equilibrium that our algorithm will find. However, the path that the algorithm will follow and the number of functional evaluations needed to achieve a given accuracy will vary drastically with the initial approximation and the degree of the subdivision chosen. For example, if we choose a subdivision of degree (10, 10), Figure 7.3 results.

The shaded triangles are the only completely labelled triangles that the algorithm will produce. (There are four other completely labelled triangles but they are incorrectly oriented.) If we start with an approximation near the center of the square (assuming ignorance this is not a bad start) we will be led to the triangle near $x = 0$, $y = 0.50$. (Indeed, the program run reported in the paper following does exactly this and takes the vertex labelled 3 as an approximate equilibrium corresponding to $x = 0$, $y = 0.4$ that is,

$$f_{11} = 0,\ f_{12} = 2,\ f_{21} = 0.8,\ f_{22} = 1.2\ )$$

The reason that the algorithm leads us to this approximation when started from the center is easily explained by examining Figure 7.4 corresponding to the artifical labelling of the square.

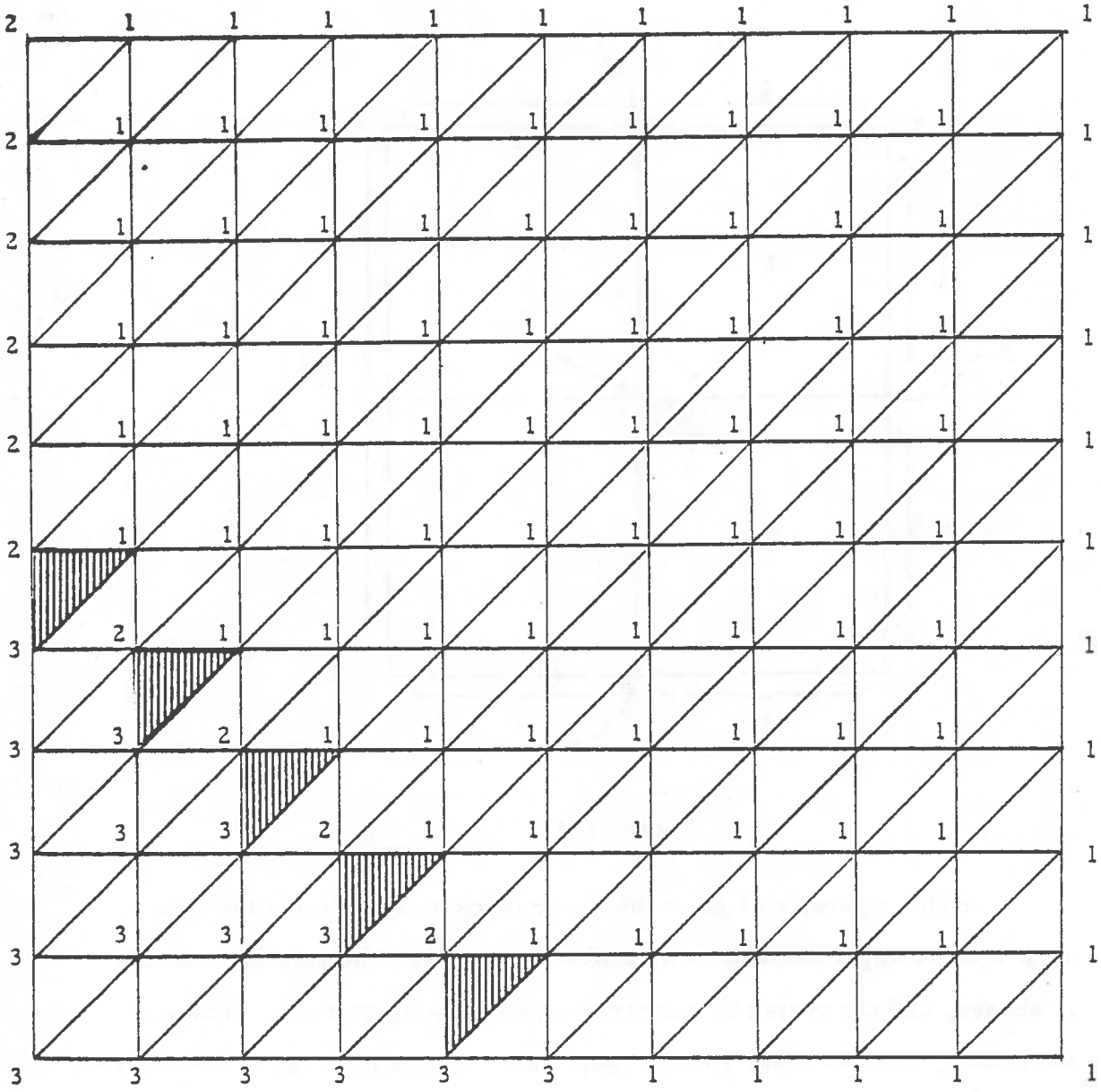


Figure 7.3

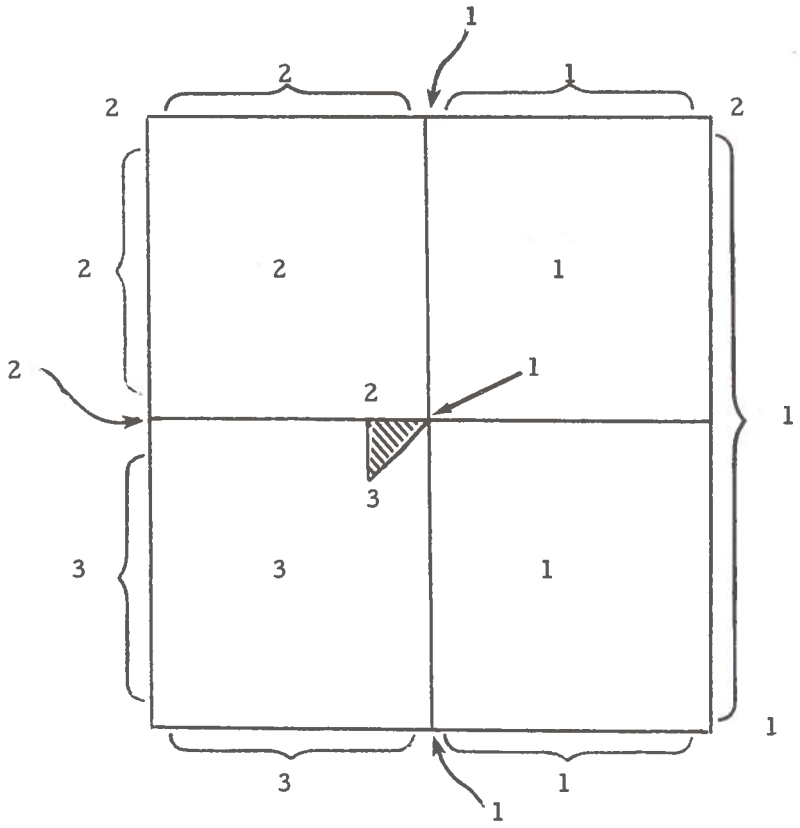


Figure 7.4

In this figure, ties go to the lower index on boundaries between open sets (of any dimension) with different indices. The starting simplex is shaded, and the vertex to complete it lies directly above the center of the square, and has label 1 (see Figure 7.2). Therefore, the vertex to leave is the center of the square and we start moving to the left.

Similar reasoning will tell us roughly the direction of movement from any starting point. Indeed, for a fine enough subdivision the path of the algorithm will be roughly as described in Figure 7.5.

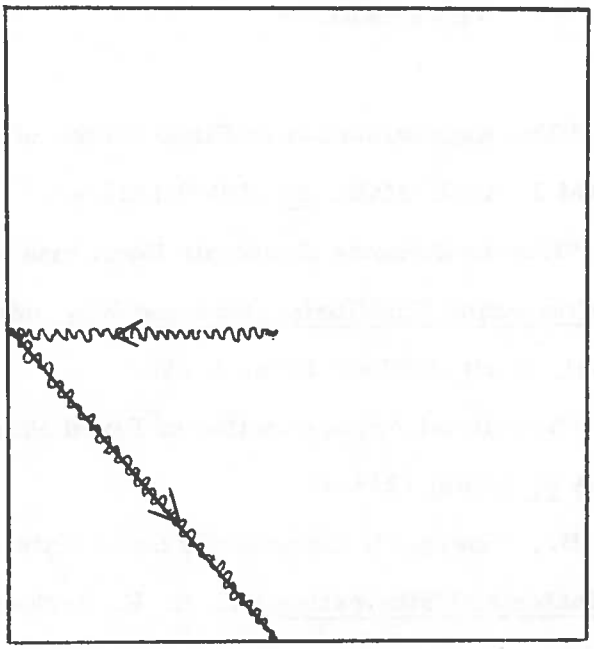


Figure 7.5

Of course, this path will be interrupted by false leads such as the shaded triangles of Figure 7.3. The efficiency of the algorithm is a trade-off between traversing very long paths at a fine mesh and being trapped by "false" completely labelled triangles at a coarse mesh. At present there is no theory to guide us in choosing the degree of the subdivision to start with and the rate at which to increase it. However, as computational experience is gathered on traffic assignment problems, we should be able to give some reliable "rules of thumb."

REFERENCES

[1] Scarf, H. E., "The Approximation of Fixed Points of a Continuous Mapping," SIAM J. Appl. Math. 15 (1967) 1328-43.

[2] Kuhn, H. W., "How to Compute Economic Equilibria by Pivotal Methods," in Computing Equilibria: How and Why (eds. J. Los and M. W. Los), North-Holland 1976, 21-38.

[3] Kuhn, H. W., "Simplicial Approximation of Fixed Point," Proc. N.A.S., U.S.A 61 (1968) 1238-42.

[4] Tompkins, C. B., "Sperner's Lemma and Some Extensions," in Applied Combinatorial Mathematics (ed. E. F. Beckenbach), John Wiley 1964.

[5] Kuhn, H. W. and Mackinnon, J. G., "Sandwich Method for Finding Fixed Points," J. Opt. Th. and Appl. 17 (1975) 189-204.

[6] Wardrop, J. G., "Some Theoretical Aspects of Road Traffic Research," Proc. Inst. of Civil Eng. (1952) 325-378.

[7] Eaves, B. C., "Homotopies for Computation of Fixed Points," Math. Prog. 3 (1972) 1-22.

[8] Merrill, O. H., "Applications and Extensions of an Algorithm that Computes Fixed Points of Certain Upper Semicontinuous Point-to-Set Mappings," Univ. of Michigan, Ph. D. Thesis, 1972.

[9] Merrill, O. H., "A Summary of Techniques for Computing Fixed Points of Continuous Mappings," in Mathematical Topics in Economic Theory and Computation, S.I.A.M., 1972.

PATHFIX: A FIXED POINT METHOD FOR COMPUTING TRAFFIC EQUILIBRIA (PROGRAM DOCUMENTATION)

1. Introduction

This report documents the program PATHFIX written by MATHTECH for the U.S. Department of Transportation as part of contract DOT-TSC-1232. In the remaining sections input formats, a sample problem, a flowchart, test problem results, and a program listing are given.

PATHFIX is a fixed point algorithm of the type originally developed by Scarf[1] for determining economic equilibria. PATHFIX, on the other hand, computes traffic equilibria (as defined by Wardrop's first principle). The theory of the algorithm is documented in the MATHTECH report "Aggregation in Network Models for Transportation Planning," (October 1975) prepared for D.O.T. under contract DOT-TSC-883, and in the first report of this volume.

As its name suggests, PATHFIX works in the space of (proportionate) path flows. As such, it differs from other procedures which work in the space of link flows. Thus the user is required to input paths between OD pairs. The requirement limits the applicability to small problems, or as a subroutine in a (restriction type) algorithm which selects from among many paths a few for positive flow. The advantage of PATHFIX is that it gives very accurate solutions (in the detail link flows) which are often much better than those that result from other methods. This is demonstrated in the three test problems (see Section 5) which were solved and

---

[1] H. E. Scarf, "The Approximation of Fixed Points of a Continuous Mapping." SIAM Journal of Applied Mathematics 15 (1967), 1328-43.

compared with other solution methods (namely, the Frank-Wolfe algorithm and quadratic programming).

Grateful acknowledgment is due Fernando Lasaga for his very capable programming of the PATHFIX algorithm.

2. PATHFIX: Input Formats

The following two pages describe the input formats for PATHFIX. These are most easily understood if studied in conjunction with the example problem of Section 3.

PATHFIX Input Format

CARD 1 PARAMETERS

cc 1-12	F12.8	CLOSE	Program stops when path costs are at most "CLOSE" apart for each OD pair.
cc 13-24	F12.8	TIME	Program prints the new pivot being considered at "TIME" seconds. Program then stops.
cc 25-36	F12.8	DINCR	This variable specifies how to refine the grid. It actually is a multiplier for the ID(I)'s.

CARD 2 NETWORK SIZE CARD:

cc 1-3	I3	Number of OD pairs.
cc 4-6	I3	Number of links.

OD PAIR DATA CARDS: (1 Card for Each OD pair)

cc 1-2	I2	Number of paths for the OD pair.
cc 3-6	I4	Demand of OD pair.

NETWORK DATA CARDS: (1 Card for Each Path)

(Put in the same order as the OD pair data cards.)

cc 1-80	80I1	If link numbered K is on path then Kth entry is 1; otherwise entry is 0. (Rows of the path-link incidence matrix.)

COST FUNCTION PARAMETERS: (1 Card for Each Link)

cc 1-4	I4	Provides input for COEFF (55,4), an INTEGER array used in computing costs and the nonlinear programming objective function.
cc 4-8	I4	
cc 9-12	I4	
cc 13-16	I4	

INITIAL APPROXIMATION CARDS (1 Card for Each OD Pair)

cc 1-78	26I3	The initial integer distributions on the paths of the OD pair.

ID CARD:

cc 1-80	10I4	Entries, in previously established order of the OD pairs, of SUM + 1 where SUM is the number of integer distributions of paths for each OD pair.

Notes:

1. ID(I) represents the integer amounts that must be distributed among the paths of OD pair I. (See program.)

2. As the program is now dimensioned it can solve problems with at most:

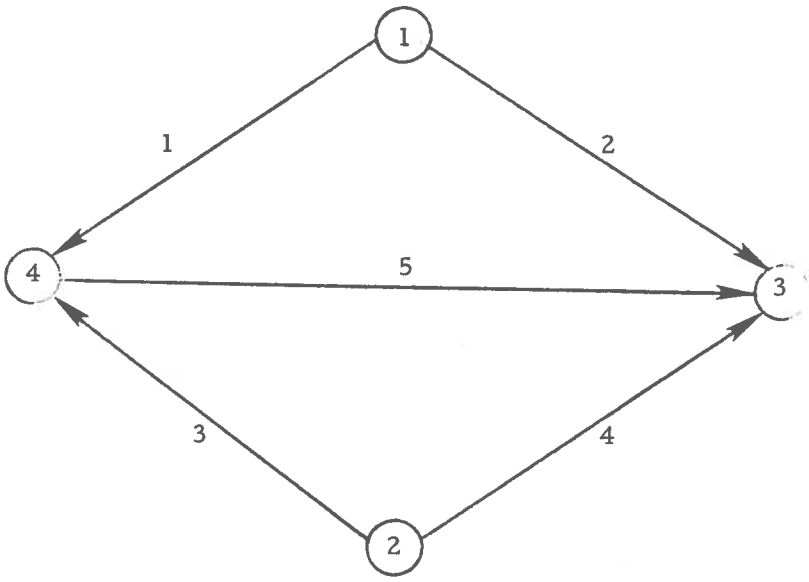
   20 OD pairs
   55 links
   50 total paths
   26 paths per OD pair.

   (The program requires 90K bytes of storage.)

3. One must input the COST and OFC functions. These use the array COEFF(50, 4). (See program.)

3. Example Problem

Consider the four node problem of the following figure:



OD Matrix:

	3
1	2
2	2

Link Times/Unit:

1 - 2
2 - 15
3 - 4
4 - 16
5 - $x + x^2$ where x is the flow on link 5.

It can be easily verified that the user equilibrium solution is:

Optimal link flows:

1 - 2
2 - 0
3 - 1
4 - 1
5 - 3

For the PATHFIX algorithm the paths are

Path	Links
1	4
2	3 - 5
3	2
4	1 - 5

The time per unit is input as $A + Bx + Cx^2$ on each link where A and B are cost function entries as given below:

Link	A	B	C
1	2	0	0
2	15	0	0
3	4	0	0
4	16	0	0
5	0	1	1

(Note that more generally there can be four parameters per link cost function.) These values are stored in the array COEFF in the program and are available to the user supplied FUNCTIONs COST and OFC. On the following pages are the inputs for this example problem and the user supplied FUNCTIONs. These are followed by the output of PATHFIX at each restart.

The strength of the method is its accuracy. At the seventeenth restart the link flows and costs are correct to seven digits. Even as early as the eleventh restart the link flows are correct to four digits. For comparison, the Frank-Wolfe algorithm does not achieve more than two digits of accuracy in fifty iterations. For other comparisons see the last section.

EXAMPLE PROBLEM INPUT

```
ACCURACY ALLOWANCE,TIME LIMIT,INCREMENT
    0.00900000 20.00000000  2.00000000
#OD PAIRS,#LINKS
    2  5
FOR EACH OD PAIR WE LIST # PATHS AND DEMAND ON OD PAIR
   2   2
   2   2
TRANSPOSE OF THE LINK-PATH INCIDENCE MATRIX:
  00010
  00101
  01000
  10001
FOR EACH LINK WE CAN INPUT FOUR ENTRIES FOR COST FUNC
     2   0   0   0
    15   0   0   0
     4   0   0   0
    16   0   0   0
     0   1   1   0
EACH LINE REPPESENTS THE INITIAL DISTRIBUTIONS ON THE PATHS OF AN OD PAIR. LAST LINE LISTS EACH THEIR SUMS PLUS ONE
    5  4
    5  4
    10  10
```

```
FORTRAN IV G LEVEL  21                  COS1                DATE = 77153          19/49/52              PAGE 0001

0001                 REAL FUNCTION COST*8(I,J)
0002                 INTEGER COEFF(55,4)
0003                 REAL*8 A,B,C,D,J,DFLOAT
0004                 COMMON/PRICE/COEFF
          C          PUT YOUR COS1 FUNCTION INBETWEEN THESE COMMENT CARDS
0005                 COST=(COEFF(I,3)*J +COEFF(I,2))*J + COEFF(I,1)
          C          ********************************************************************************
0006          3      RETURN
0007                 END
```

```
FORTRAN IV G LEVEL  21                  OPC             DATE = 77153        19/49/52            PAGE 0001

 0001                REAL FUNCTION OPC*8(I,J)
 0002                INTEGER COEFF(55,4)
 0003                REAL*8 A,B,C,D,J,DFLOAT
 0004                REAL*8 DLOG
 0005                COMMON/PRICE/COEFF
                C    PUT YOUR NONLIN. PRO. OBJ. FUNC. INBETWEEN THESE COMMENT CARDS
 0006                OPC=((COEFF(I,3)*J/3.0D0*COEFF(I,2)/2.0D0)*J + COEFF(I,1))*J
                C    ********************************************************************************
 0007           3    RETURN
 0008                END
```

EXAMPLE PROBLEM OUTPUT

```
RESTART NUMBER:  1
OD PAIR  1FLOWS:     0.0           2.00000000
PATHCOSTS:    16.00000000   17.44000000
OD PAIR  2FLOWS:     0.80000000    1.20000000
PATHCOSTS:    15.00000000   15.44000000
MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:       0.44000000
LINK  1 HAS FLOW     1.200000 AT COST         2.000000
LINK  2 HAS FLOW     0.800000 AT COST        15.000000
LINK  3 HAS FLOW     2.000000 AT COST         4.000000
LINK  4 HAS FLOW     0.0      AT COST        16.000000
LINK  5 HAS FLOW     3.200000 AT COST        13.440000
ARTIFICIAL EVA.          8FUNC.EVA.          7
TOTAL COST IF PLAN IS IMPLEMENTED:      65.40800000
NONLINEAR PROGRAMMING OBJ.FUNC.      38.44266319
TIME UP TO THIS RESTART:     0.04939270MAXIMUM SUM OF Z          23
************************************************** ****************************************

RESTART NUMBER:  2
OD PAIR  1FLOWS:     0.95652174    1.04347826
PATHCOSTS:    16.00000000   15.69754253
OD PAIR  2FLOWS:     0.08695652    1.91304348
PATHCOSTS:    15.00000000   13.69754253
MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:       1.30245747
LINK  1 HAS FLOW     1.913043 AT COST         2.000000
LINK  2 HAS FLOW     0.086957 AT COST        15.000000
LINK  3 HAS FLOW     1.043478 AT COST         4.000000
LINK  4 HAS FLOW     0.956522 AT COST        16.000000
LINK  5 HAS FLOW     2.956522 AT COST        11.697543
ARTIFICIAL EVA.          28FUNC.EVA.         48
TOTAL COST IF PLAN IS IMPLEMENTED:      59.19273445
NONLINEAR PROGRAMMING OBJ.FUNC.      37.59354210
TIME UP TO THIS RESTART:     0.08985901MAXIMUM SUM OF Z          53
************************************************** ****************************************

RESTART NUMBER:  3
OD PAIR  1FLOWS:     0.98113208    1.01886792
PATHCOSTS:    16.00000000   15.86828053
OD PAIR  2FLOWS:     0.03773585    1.96226415
PATHCOSTS:    15.00000000   13.86828053
MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:       1.13171947
LINK  1 HAS FLOW     1.962264 AT COST         2.000000
LINK  2 HAS FLOW     0.037736 AT COST        15.000000
LINK  3 HAS FLOW     1.018868 AT COST         4.000000
LINK  4 HAS FLOW     0.981132 AT COST        16.000000
LINK  5 HAS FLOW     2.981132 AT COST        11.968281
ARTIFICIAL EVA.          31FUNC.EVA.         55
TOTAL COST IF PLAN IS IMPLEMENTED:      59.64506270
NONLINEAR PROGRAMMING OBJ.FUNC.      37.53897721
TIME UP TO THIS RESTART:     0.10223389MAXIMUM SUM OF Z         122
************************************************** ****************************************

RESTART NUMBER:  4
OD PAIR  1FLOWS:     0.98360656    1.01639344
PATHCOSTS:    16.00000000   16.00000000
OD PAIR  2FLOWS:     0.01639344    1.98360656
PATHCOSTS:    15.00000000   14.00000000
MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:       1.00000000
LINK  1 HAS FLOW     1.983607 AT COST         2.000000
LINK  2 HAS FLOW     0.016393 AT COST        15.000000
LINK  3 HAS FLOW     1.016393 AT COST         4.000000
LINK  4 HAS FLOW     0.983607 AT COST        16.000000
LINK  5 HAS FLOW     3.000000 AT COST        12.000000
```

```
ARTIFICIAL EVA.        33FUNC.EVA.        60
TOTAL COST IF PLAN IS IMPLEMENTED:     60.01639344
NONLINEAR PROGRAMMING OBJ.FUNC.     37.51639128
TIME UP TO THIS RESTART:     0.11439514MAXIMUM SUM OF Z         281
**************************************************** ****************************************
RESTART NUMBER:  5
OD PAIR  1FLOWS:     0.99644128     1.00355872
PATHCOSTS:    16.00000000    15.97510163
OD PAIR  2FLOWS:     0.00711744     1.99288256
PATHCOSTS:    15.00000000    13.97510163
MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:       1.02489837
LINK  1 HAS FLOW     1.992883 AT COST         2.000000
LINK  2 HAS FLOW     0.007117 AT COST        15.000000
LINK  3 HAS FLOW     1.003559 AT COST         4.000000
LINK  4 HAS FLOW     0.996441 AT COST        16.000000
LINK  5 HAS FLOW     2.996441 AT COST        11.975102
ARTIFICIAL EVA.        37FUNC.EVA.        69
TOTAL COST IF PLAN IS IMPLEMENTED:     59.98251094
NONLINEAR PROGRAMMING OBJ.FUNC.     37.50715977
TIME UP TO THIS RESTART:     0.12937927MAXIMUM SUM OF Z         648
**************************************************** ****************************************
RESTART NUMBER:  6
OD PAIR  1FLOWS:     0.99691358     1.00308642
PATHCOSTS:    16.00000000    16.00000000
OD PAIR  2FLOWS:     0.00306642     1.99691358
PATHCOSTS:    15.00000000    14.00000000
MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:       1.00000000
LINK  1 HAS FLOW     1.996914 AT COST         2.000000
LINK  2 HAS FLOW     0.003086 AT COST        15.000000
LINK  3 HAS FLOW     1.003086 AT COST         4.000000
LINK  4 HAS FLOW     0.996914 AT COST        16.000000
LINK  5 HAS FLOW     3.000000 AT COST        12.000000
ARTIFICIAL EVA.        39FUNC.EVA.        74
TOTAL COST IF PLAN IS IMPLEMENTED:     60.00308642
NONLINEAR PROGRAMMING OBJ.FUNC.     37.50308436
TIME UP TO THIS RESTART:     0.14039612MAXIMUM SUM OF Z        1495
**************************************************** ****************************************
RESTART NUMBER:  7
OD PAIR  1FLOWS:     0.99933110     1.00066890
PATHCOSTS:    16.00000000    15.99531817
OD PAIR  2FLOWS:     0.00133779     1.99866221
PATHCOSTS:    15.00000000    13.99531817
MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:       1.00468183
LINK  1 HAS FLOW     1.998662 AT COST         2.000000
LINK  2 HAS FLOW     0.001338 AT COST        15.000000
LINK  3 HAS FLOW     1.000669 AT COST         4.000000
LINK  4 HAS FLOW     0.999331 AT COST        16.000000
LINK  5 HAS FLOW     2.999331 AT COST        11.995318
ARTIFICIAL EVA.        43FUNC.EVA.        83
TOTAL COST IF PLAN IS IMPLEMENTED:     59.99729544
NONLINEAR PROGRAMMING OBJ.FUNC.     37.50133672
TIME UP TO THIS RESTART:     0.15484619MAXIMUM SUM OF Z        3449
**************************************************** ****************************************
RESTART NUMBER:  8
OD PAIR  1FLOWS:     0.99971006     1.00028994
PATHCOSTS:    16.00000000    15.99797051
OD PAIR  2FLOWS:     0.00057988     1.99942012
PATHCOSTS:    15.00000000    13.99797051
MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:       1.00202949
LINK  1 HAS FLOW     1.999420 AT COST         2.000000
```

```
LINK  2 HAS FLOW     0.000586 AT COST        15.000000
LINK  3 HAS FLOW     1.000290 AT COST         4.000000
LINK  4 HAS FLOW     0.999710 AT COST        16.000000
LINK  5 HAS FLOW     2.999710 AT COST        11.999971
ARTIFICIAL EVA.         46FUNC.EVA.         90
TOTAL COST IF PLAN IS IMPLEMENTED:      59.99949200
NONLINEAR PROGRAMMING OBJ.FUNC.      37.50057859
TIME UP TO THIS RESTART:     0.16827393MAXIMUM SUM OF Z         7957
**************************************************** ****************************************
RESTART NUMBER:  9
OD PAIR  1FLOWS:     0.99987432     1.00012568
PATHCOSTS:    16.00000000    15.99912029
OD PAIR  2FLOWS:     0.00025135     1.99974865
PATHCOSTS:    15.00000000    13.99912029
MAXIMUM DIFFEPENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:       1.00087971
LINK  1 HAS FLOW     1.999749 AT COST         2.000000
LINK  2 HAS FLOW     0.000251 AT COST        15.000000
LINK  3 HAS FLOW     1.000126 AT COST         4.000000
LINK  4 HAS FLOW     0.999874 AT COST        16.000000
LINK  5 HAS FLOW     2.999874 AT COST        11.999120
ARTIFICIAL EVA.         49FUNC.EVA.         97
TOTAL COST IF PLAN IS IMPLEMENTED:      59.99761232
NONLINEAR PROGRAMMING OBJ.FUNC.      37.50024930
TIME UP TO THIS RESTART:     0.18095398MAXIMUM SUM OF Z        18357
**************************************************** ****************************************
RESTART NUMBER: 10
OD PAIR  1FLOWS:     0.99994552     1.00005448
PATHCOSTS:    16.00000000    15.99961868
OD PAIR  2FLOWS:     0.00010895     1.99989105
PATHCOSTS:    15.00000000    13.99961868
MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:       1.00038132
LINK  1 HAS FLOW     1.999891 AT COST         2.000000
LINK  2 HAS FLOW     0.000109 AT COST        15.000000
LINK  3 HAS FLOW     1.000054 AT COST         4.000000
LINK  4 HAS FLOW     0.999946 AT COST        16.000000
LINK  5 HAS FLOW     2.999946 AT COST        11.999619
ARTIFICIAL EVA.         52FUNC.EVA.        104
TOTAL COST IF PLAN IS IMPLEMENTED:      59.99896500
NONLINEAR PROGRAMMING OBJ.FUNC.      37.50010742
TIME UP TO THIS RESTART:     0.19354248MAXIMUM SUM OF Z        42350
**************************************************** ****************************************
RESTART NUMBER: 11
OD PAIR  1FLOWS:     1.00000000     1.00000000
PATHCOSTS:    16.00000000    15.99966942
OD PAIR  2FLOWS:     0.00004723     1.99995277
PATHCOSTS:    15.00000000    13.99966942
MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:       1.00033058
LINK  1 HAS FLOW     1.999953 AT COST         2.000000
LINK  2 HAS FLOW     0.000047 AT COST        15.000000
LINK  3 HAS FLOW     1.000000 AT COST         4.000000
LINK  4 HAS FLOW     1.000000 AT COST        16.000000
LINK  5 HAS FLOW     2.999953 AT COST        11.999669
ARTIFICIAL EVA.         55FUNC.EVA.        111
TOTAL COST IF PLAN IS IMPLEMENTED:      59.99905551
NONLINEAR PROGRAMMING OBJ.FUNC.      37.50004558
TIME UP TO THIS RESTART:     0.20591736MAXIMUM SUM OF Z        97702
**************************************************** ****************************************
RESTART NUMBER: 12
OD PAIR  1FLOWS:     0.99997953     1.00002047
PATHCOSTS:    16.00000000    16.00000000
```

```
 OD PAIR  2FLOWS:    0.00002047    1.99997953
 PATHCOSTS:    15.00000000   14.00000000
 MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:      1.00000000
 LINK  1 HAS FLOW    1.999980 AT COST        2.000000
 LINK  2 HAS FLOW    0.000020 AT COST       15.000000
 LINK  3 HAS FLOW    1.000020 AT COST        4.000000
 LINK  4 HAS FLOW    0.999980 AT COST       16.000000
 LINK  5 HAS FLOW    3.000000 AT COST       12.000000
 ARTIFICIAL EVA.        56FUNC.EVA.       114
 TOTAL COST IF PLAN IS IMPLEMENTED:     60.00002047
 NONLINEAR PROGRAMMING OBJ.FUNC.     37.50001810
 TIME UP TO THIS RESTART:    0.21537781MAXIMUM SUM OF Z      225399
 ************************************************** ****************************************
 RESTART NUMBER: 13
 OD PAIR  1FLOWS:    0.99999556    1.00000444
 PATHCOSTS:    16.00000000   15.99996894
 OD PAIR  2FLOWS:    0.00000887    1.99999113
 PATHCOSTS:    15.00000000   13.99996894
 MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:      1.00003106
 LINK  1 HAS FLOW    1.999991 AT COST        2.000000
 LINK  2 HAS FLOW    0.000009 AT COST       15.000000
 LINK  3 HAS FLOW    1.000004 AT COST        4.000000
 LINK  4 HAS FLOW    0.999996 AT COST       16.000000
 LINK  5 HAS FLOW    2.999996 AT COST       11.999969
 ARTIFICIAL EVA.        60FUNC.EVA.       123
 TOTAL COST IF PLAN IS IMPLEMENTED:     59.99991570
 NONLINEAR PROGRAMMING OBJ.FUNC.     37.50000721
 TIME UP TO THIS RESTART:    0.22921753MAXIMUM SUM OF Z      519996
 ************************************************** ****************************************
 RESTART NUMBER: 14
 OD PAIR  1FLOWS:    0.99999615    1.00000385
 PATHCOSTS:    16.00000000   16.00000000
 OD PAIR  2FLOWS:    0.00000385    1.99999615
 PATHCOSTS:    15.00000000   14.00000000
 MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:      1.00000000
 LINK  1 HAS FLOW    1.999996 AT COST        2.000000
 LINK  2 HAS FLOW    0.000004 AT COST       15.000000
 LINK  3 HAS FLOW    1.000004 AT COST        4.000000
 LINK  4 HAS FLOW    0.999996 AT COST       16.000000
 LINK  5 HAS FLOW    3.000000 AT COST       12.000000
 ARTIFICIAL EVA.        62FUNC.EVA.       128
 TOTAL COST IF PLAN IS IMPLEMENTED:     60.00000385
 NONLINEAR PROGRAMMING OBJ.FUNC.     37.50000143
 TIME UP TO THIS RESTART:    0.24012756MAXIMUM SUM OF Z     1199631
 ************************************************** ****************************************
 RESTART NUMBER: 15
 OD PAIR  1FLOWS:    0.99999917    1.00000083
 PATHCOSTS:    16.00000000   15.99999416
 OD PAIR  2FLOWS:    0.00000167    1.99999833
 PATHCOSTS:    15.00000000   13.99999416
 MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:      1.00000584
 LINK  1 HAS FLOW    1.999998 AT COST        2.000000
 LINK  2 HAS FLOW    0.000002 AT COST       15.000000
 LINK  3 HAS FLOW    1.000001 AT COST        4.000000
 LINK  4 HAS FLOW    0.999999 AT COST       16.000000
 LINK  5 HAS FLOW    2.999999 AT COST       11.999994
 ARTIFICIAL EVA.        66FUNC.EVA.       137
 TOTAL COST IF PLAN IS IMPLEMENTED:     59.99999416
 NONLINEAR PROGRAMMING OBJ.FUNC.     37.50100021
 TIME UP TO THIS RESTART:    0.25407410MAXIMUM SUM OF Z     2767549
```

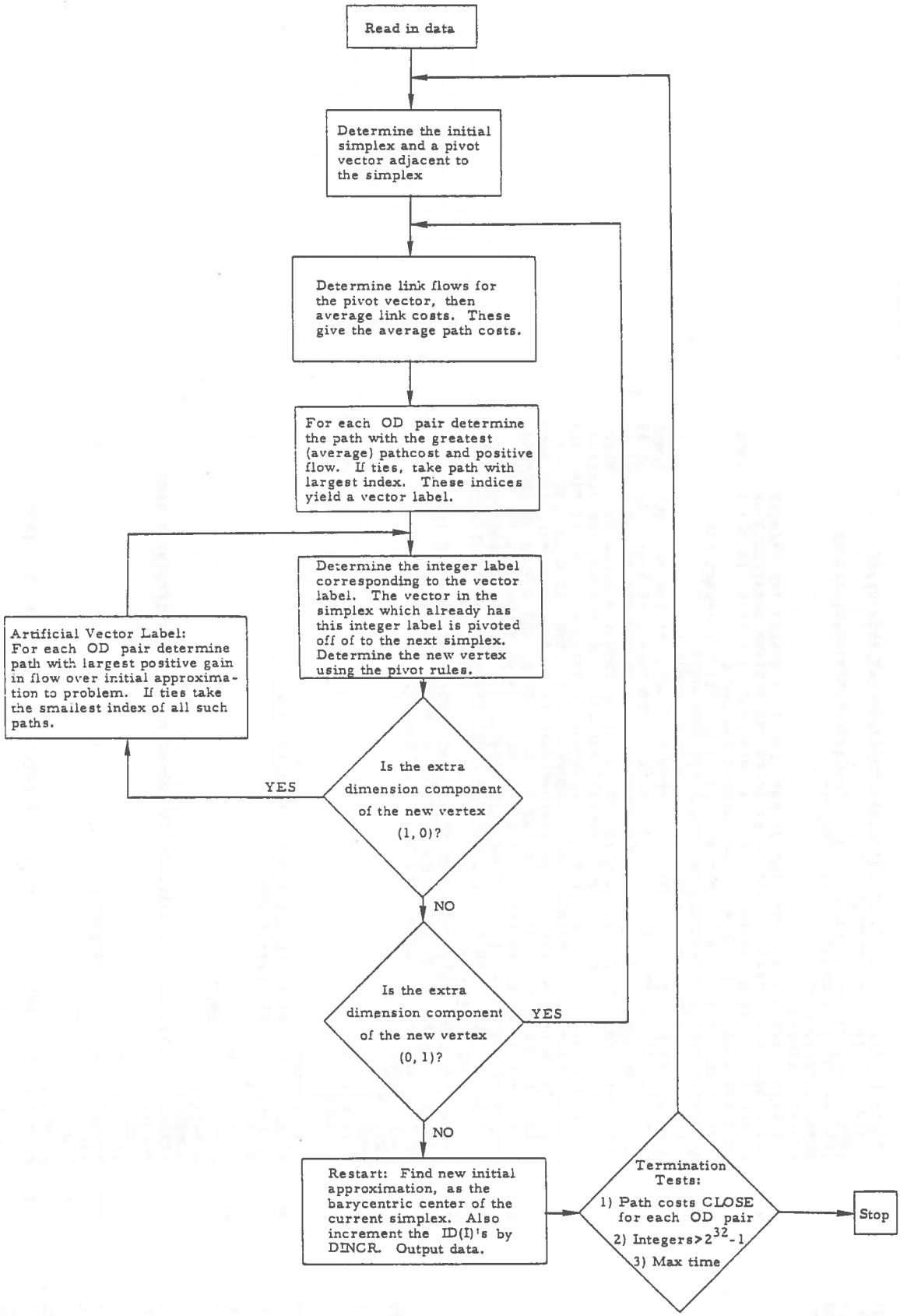
```
**************************************************** ******************************************
RESTART NUMBER: 16
OD PAIR  1FLOWS:    0.99999964    1.00000036
PATHCOSTS:   16.00000000   15.99999747
OD PAIR  2FLOWS:    0.00000072    1.99999928
PATHCOSTS:   15.00000000   13.99999747
MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:      1.00000253
LINK  1 HAS FLOW    1.999999 AT COST        2.000000
LINK  2 HAS FLOW    0.000001 AT COST       15.000000
LINK  3 HAS FLOW    1.000000 AT COST        4.000000
LINK  4 HAS FLOW    1.000000 AT COST       16.000000
LINK  5 HAS FLOW    3.000000 AT COST       11.999997
ARTIFICIAL EVA.       69FUNC.EVA.      144
TOTAL COST IF PLAN IS IMPLEMENTED:     59.99999313
NONLINEAR PROGRAMMING OBJ.FUNC.     37.49999844
TIME UP TO THIS RESTART:    0.26727295MAXIMUM SUM OF Z     6384736
**************************************************** ******************************************
RESTART NUMBER: 17
OD PAIR  1FLOWS:    0.99999969    1.00000031
PATHCOSTS:   16.00000000   16.00000000
OD PAIR  2FLOWS:    0.00000031    1.99999969
PATHCOSTS:   15.00000000   14.00000000
MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:      1.00000000
LINK  1 HAS FLOW    2.000000 AT COST        2.000000
LINK  2 HAS FLOW    0.000000 AT COST       15.000000
LINK  3 HAS FLOW    1.000000 AT COST        4.000000
LINK  4 HAS FLOW    1.000000 AT COST       16.000000
LINK  5 HAS FLOW    3.000000 AT COST       12.000000
ARTIFICIAL EVA.       71FUNC.EVA.      149
TOTAL COST IF PLAN IS IMPLEMENTED:     60.00000031
NONLINEAR PROGRAMMING OBJ.FUNC.     37.49999802
TIME UP TO THIS RESTART:    0.27839661MAXIMUM SUM OF Z    14729586
**************************************************** ******************************************
RESTART NUMBER: 18
OD PAIR  1FLOWS:    0.99999986    1.00000014
PATHCOSTS:   16.00000000   16.00000000
OD PAIR  2FLOWS:    0.00000014    1.99999986
PATHCOSTS:   15.00000000   14.00000000
MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:      1.00000000
LINK  1 HAS FLOW    2.000000 AT COST        2.000000
LINK  2 HAS FLOW    0.000000 AT COST       15.000000
LINK  3 HAS FLOW    1.000000 AT COST        4.000000
LINK  4 HAS FLOW    1.000000 AT COST       16.000000
LINK  5 HAS FLOW    3.000000 AT COST       12.000000
ARTIFICIAL EVA.       74FUNC.EVA.      156
TOTAL COST IF PLAN IS IMPLEMENTED:     60.00000014
NONLINEAR PROGRAMMING OBJ.FUNC.     37.49999727
TIME UP TO THIS RESTART:    0.29119873MAXIMUM SUM OF Z    33981155
**************************************************** ******************************************
RESTART NUMBER: 19
OD PAIR  1FLOWS:    0.99999997    1.00000003
PATHCOSTS:   16.00000000   15.99999979
OD PAIR  2FLOWS:    0.00000006    1.99999994
PATHCOSTS:   15.00000000   13.99999979
MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:      1.00000021
LINK  1 HAS FLOW    2.000000 AT COST        2.000000
LINK  2 HAS FLOW    0.000000 AT COST       15.000000
LINK  3 HAS FLOW    1.000000 AT COST        4.000000
LINK  4 HAS FLOW    1.000000 AT COST       16.000000
LINK  5 HAS FLOW    3.000000 AT COST       12.000000
```

```
ARTIFICIAL EVA.        7cFUNC.EVA.        165
TOTAL COST IF PLAN IS IMPLEMENTED:     59.99999988
NONLINEAR PROGRAMMING OBJ.FUNC.     37.49999802
TIME UP TO THIS RESTART:     0.30575562MAXIMUM SUM OF Z    78394524
************************************************** ******************************************
RESTART NUMBER: 20
OD PAIR  1FLOWS:     0.99999997     1.00000003
PATHCOSTS:    16.00000000    16.00000000
OD PAIR  2FLOWS:     0.00000003     1.99999997
PATHCOSTS:    15.00000000    14.00000000
MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:       1.00000000
LINK  1 HAS FLOW     2.000000 AT COST         2.000000
LINK  2 HAS FLOW     0.000000 AT COST        15.000000
LINK  3 HAS FLOW     1.000000 AT COST         4.000000
LINK  4 HAS FLOW     1.000000 AT COST        16.000000
LINK  5 HAS FLOW     3.000000 AT COST        12.000000
ARTIFICIAL EVA.        80FUNC.EVA.        170
TOTAL COST IF PLAN IS IMPLEMENTED:     60.00000003
NONLINEAR PROGRAMMING OBJ.FUNC.     37.49999752
TIME UP TO THIS RESTART:     0.31730652MAXIMUM SUM OF Z    180856167
************************************************** ******************************************
RESTART NUMBER: 21
OD PAIR  1FLOWS:     0.99999999     1.00000001
PATHCOSTS:    16.00000000    15.99999996
OD PAIR  2FLOWS:     0.00000001     1.99999999
PATHCOSTS:    15.00000000    13.99999996
MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:       1.00000004
LINK  1 HAS FLOW     2.000000 AT COST         2.000000
LINK  2 HAS FLOW     0.000000 AT COST        15.000000
LINK  3 HAS FLOW     1.000000 AT COST         4.000000
LINK  4 HAS FLOW     1.000000 AT COST        16.000000
LINK  5 HAS FLOW     3.000000 AT COST        12.000000
ARTIFICIAL EVA.        84FUNC.EVA.        179
TOTAL COST IF PLAN IS IMPLEMENTED:     59.99999989
NONLINEAR PROGRAMMING OBJ.FUNC.     37.49999730
TIME UP TO THIS RESTART:     0.33145142MAXIMUM SUM OF Z    417235177
************************************************** ******************************************
RESTART NUMBER: 22
OD PAIR  1FLOWS:     1.00000000     1.00000000
PATHCOSTS:    16.00000000    15.99999998
OD PAIR  2FLOWS:     0.00000000     2.00000000
PATHCOSTS:    15.00000000    13.99999998
MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:       1.00000002
LINK  1 HAS FLOW     2.000000 AT COST         2.000000
LINK  2 HAS FLOW     0.000000 AT COST        15.000000
LINK  3 HAS FLOW     1.000000 AT COST         4.000000
LINK  4 HAS FLOW     1.000000 AT COST        16.000000
LINK  5 HAS FLOW     3.000000 AT COST        12.000000
ARTIFICIAL EVA.        87FUNC.EVA.        186
TOTAL COST IF PLAN IS IMPLEMENTED:     59.99999995
NONLINEAR PROGRAMMING OBJ.FUNC.     37.49999721
TIME UP TO THIS RESTART:     0.34454346MAXIMUM SUM OF Z    962561553
************************************************** ******************************************
RESTART NUMBER: 23
OD PAIR  1FLOWS:     1.00000000     1.00000000
PATHCOSTS:    16.00000000    15.99999999
OD PAIR  2FLOWS:     0.00000000     2.00000000
PATHCOSTS:    15.00000000    13.99999999
MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIFFERENCE:       1.00000001
LINK  1 HAS FLOW     2.000000 AT COST         2.000000
```

```
LINK  2 HAS FLOW     0.000000 AT COST        15.000000
LINK  3 HAS FLOW     1.000000 AT COST         4.000000
LINK  4 HAS FLOW     1.000000 AT COST        16.000000
LINK  5 HAS FLOW     3.000000 AT COST        12.000000
ARTIFICIAL EVA.         90FUNC.EVA.        193
TOTAL COST IF PLAN IS IMPLEMENTED:      59.99999999
NONLINEAR PROGRAMMING OBJ.FUNC.      37.49999717
TIME UP TO THIS RESTART:     0.35754395MAXIMUM SUM OF Z           0
************************************************** **********************************************
INTEGER VARIABLE BIGGER THAN 2 BILLION
CUMULATIVE TIME:       0.35858154
```

4. Flowchart and Program Listing

PATHFIX FLOW CHART



```
0001              REAL*8 F(55),C(50),COST,FCOST(55),DMX,DMAX1,DELCAT,RAT(50),OF
0002              REAL*8 DNX,DOX,TIME,DINCR,CLOSE
0003              INTEGER P,Q,N(20),M(21),D(20),A(55,50),L(20),COEFF(55,4),ID(20)
0004              INTEGER Z(50,52),ZOR(52),LABEL(50),MAXO
0005              COMMON/PRICE/COEFF
                C     PROGRAM PATHFIX
                C     DNX WILL CONTAIN THE TOTAL COST IF THE PLAN IS ACTUALLY IMPLEMENTED.
                C     DOX MEASURES THE MAXIMUM OVER OD PAIRS OF THE MAXIMUM DIFFERENCE IN
                C     PATHCOSTS FOR PATHS IN AN OD PAIR THAT HAVE POSITIVE FLOW. IF DOX IS LESS
                C     THAN THE VARIABLE CLOSE THEN THE PROGRAM TERMINATES.
                C     TIME HOLDS THE TIME IN SECONDS AFTERWHICH ALL PIVOT ELEMENTS ARE
                C     OUTPUTTED WITH PROPORTIONAL PATHFLOWS AND PATHCOSTS.
                C     DINCR IS THE MULTIPLE BY WHICH OUR GRID IS MADE FINER.
                C     F HOLDS THE LINK FLOWS. C HOLDS THE PATHCOSTS. COST IS THE FUNCTION. FCOST
                C     HOLDS THE LINK COSTS. RAT HOLDS THE ACTUAL PATH FLOWS. OF WILL HOLD THE VA
                C     LUE OF OUR NONLIN. PRO. OBJECTIVE FUNCTION. OFC IS THE FUNCTION FOR OF.
                C     P IS THE NUMBER OF OD PAIRS. Q IS THE NUMBER OF LINKS. N HOLDS THE NUMBER
                C     OF PATHS PER OD PAIR. M IS A CUMULATIVE HOLDING OF N. D HOLDS THE DEMANDS
                C     ON THE OD PAIRS. A IS OUR LINK-PATH INCIDENCE MATRIX. L IS OUR VECTOR LABE
                C     L. COEFF IS USED TO COMPUTE THE FUNCTIONS COST AND OFC. ID HOLDS THE SUM
                C     FOR EACH OD PAIR OF THE INTEGER AMOUNT THAT IS DISTRIBUTED AMONG THE PATH
                C     S IN THE MATRIX Z .  THE MATRIX HAS AS MANY ROWS AS THE NUMBER OF VECTORS
                C     IN A SIMPLEX IN THE PROBLEM PLUS ONE.IT HAS AS MANY COLUMNS AS THE TOTAL
                C     NUMBER OF PATHS PLUS TWO FOR THE EXTRA DIMENSION. ZOR IS THE VECTOR OF OUR
                C     INITIAL APPROXIMATION TO THE SOLUTION. LABEL KEEPS TRACK OF WHICH ROW IN Z
                C     IS LABELLED WHICH LABEL.
                C     CANSTA AND INDUMP ARE PRINCETON UNIV. COMP. CENTER LOCAL ERROR ROUTINES.
                C     CUTIME IS A PRINCETON UNIV. COMP. CENTER SUPPLIED FUNCTION.
                C     WRITTEN BY FERNANDO LASAGA FOR MATHTECH  JUNE 2,1977
0006              CALL CANSTA
0007              CALL INDUMP
0008              IREST=0
0009              WXY=CUTIME(0)
0010              IFLAG=0
0011              IFLOG=0
0012              WRITE(6,209)
0013          209 FORMAT(' ACCURACY ALLOWANCE,TIME LIMIT,INCREMENT')
0014              READ(5,204) CLOSE,TIME,DINCR
0015          204 FORMAT(3F12.0)
0016          220 FORMAT(3X,3F12.8)
0017              WRITE(6,220) CLOSE,TIME,DINCR
0018              NUMBER=0
0019              NUMBE=0
                C     NUMBER IS TOTAL OF TRUE EVALUATIONS. NUMBE IS TOTAL ARTIFICIAL EVALUA.
0020              READ(5,1) P,Q
0021          1   FORMAT(2I3)
0022              WRITE(6,200)
0023          200 FORMAT(' #OD PAIRS,#LINKS')
0024              WRITE(6,211) P,Q
0025          211 FORMAT(3X,2I3)
0026              WRITE(6,203)
0027          203 FORMAT(' FOR EACH OD PAIR WE LIST # PATHS AND DEMAND ON OD PAIR')
0028              DO 2 I=1,P
0029              READ(5,3) N(I),D(I)
0030          3   FORMAT(I2,I4)
0031              WRITE(6,212) N(I),D(I)
```

```
0032        212   FORMAT(3X,I2,I4)
0033        2     CONTINUE
0034              M(1)=0
0035              DO 5 J=1,P
0036        5     M(J+1)=M(J)+N(I)
            C     ISN IS THE NUMBER OF VECTORS IN A SIMPLEX PLUS ONE.
0037              ISN=M(P+1) - P + 2
0038              N1=M(P+1)+2
0039              N2=M(P+1)
            C     N2 IS TOTAL # PATHS. N1=N2 + 2
0040              WRITE(6,201)
0041        201   FORMAT(' TRANSPOSE OF THE LINK-PATH INCIDENCE MATRIX:')
0042              DO 4 I=1,N2
0043              READ(5,6) (A(IM,I),IM=1,Q)
0044        6     FORMAT(80I1)
0045              WRITE(6,210) (A(IM,I),IM=1,Q)
0046        210   FORMAT(3X,80I1)
0047        4     CONTINUE
0048              WRITE(6,208)
0049        208   FORMAT(' FOR EACH LINK WE CAN INPUT FOUR ENTRIES FOR COST FUNC')
0050              DO 206 I=1,Q
0051              READ(5,205) (COEFF(I,IM),IM=1,4)
0052        205   FORMAT(4I4)
0053              WRITE(6,213) (COEFF(I,IM),IM=1,4)
0054        213   FORMAT(3X,4I4)
0055        206   CONTINUE
0056              WRITE(6,202)
0057        202   FORMAT(' EACH LINE REPRESENTS THE INITIAL DISTRIBUTIONS ON THE PA
                  1THS OF AN OD PAIR. LAST LINE LISTS EACH THEIR SUMS PLUS ONE')
0058              DO 10 I=1,P
0059              N3=M(I)+1
0060              N4=M(I+1)
0061              READ(5,13) (ZOR(J),J=N3,N4)
0062        13    FORMAT(26I3)
0063              WRITE(6,214) (ZOR(J),J=N3,N4)
0064        214   FORMAT(3X,26I3)
0065        10    CONTINUE
0066              ZOR(M(P+1)+1)=1
0067              ZOR(M(P+1)+2)=0
0068              READ(5,27) (ID(I),I=1,P)
0069        27    FORMAT(20I4)
0070              WRITE(6,216) (ID(I),I=1,P)
0071        216   FORMAT(3X,20I4)
0072              WRITE(6,217)
0073        217   FORMAT('1')
            C     ********************************************************************
            C     WE NOW COMPUTE THE INITIAL SIMPLEX AND THE NEW PIVOT ELEMENT TO START
            C     THE ALGORITHM.
0074        18    DO 11 J=1,N1
0075              DO 12 I=1,ISN
0076        12    Z(I,J) = ZOR(J)
0077        11    CONTINUE
0078              N3=ISN-1
            C     WE INITIALIZE THE LABEL ARRAY TO CORRESPOND TO THE KNOWN INTEGER LABELS.
0079              DO 7 I=1,N3
0080        7     LABEL(I)=I
```

```
0081              DO 14 I=1,P
0082              J=M(I) - I + 2
0093              K=0
0084              IJ=M(I)+1
0085              IF(M(I+1)-IJ.EQ.0) GO TO 9
0086              DO 15 IK=1,J
0087              K=K+1
0088         15   Z(K,IJ) = Z(K,IJ) + 1
0089              N3=N(I)
0090              DO 16 IK=2,N3
0091              IJ=IJ+1
0092              K=K+1
0093         16   Z(K,IJ)=Z(K,IJ) + 1
0094              N3=K+1
0095              DO 17 IK=N3,ISN
0096         17   Z(IK,IJ)=Z(IK,IJ)+1
0097              GO TO 14
0098         9    DO 8 IJN=1,ISN
0099         8    Z(IJN,IJ)=Z(IJN,IJ)+1
0100         14   CONTINUE
0101              Z(ISN,M(P+1)+1)=0
0102              Z(ISN,M(P+1)+2)=1
0103              NO=ISN
             C    WE HAVE NOW FINISHED COMPUTING THE INITIAL SIMPLEX. NO INDEXES THE ROW
             C    IN Z CONTAINING OUR NEW PIVOT ELEMENT.
             C    ********************************************************************************
             C    WE GO TO LINE 19 WHEN WE WANT TO COMPUTE A REAL VECTOR LABELLING.
0104         19   NUMBER=NUMBER+1
0105              DO 21 J=1,P
0106              N3=M(J)+1
0107              N4=M(J+1)
0108              DO 22 K=N3,N4
0109         22   RAT(K)=DFLOAT(Z(NO,K))*DFLOAT(D(J))/DFLOAT(ID(J))
0110         21   CONTINUE
0111              DO 20 I=1,Q
0112              F(I)=0
0113              DO 25 J=1,N2
0114              IF(A(I,J).EQ.0) GO TO 25
0115              F(I)=F(I)+RAT(J)
0116         25   CONTINUE
0117              FCOST(I) = COST(I,F(I))
0118         20   CONTINUE
0119              DO 23 I=1,N2
0120              C(I)=0
0121              DO 24 J=1,Q
0122              IF(A(J,I).EQ.0) GO TO 24
0123              C(I) = C(I) + FCOST(J)
0124         24   CONTINUE
0125         23   CONTINUE
             C    THE DO LOOP INDEXED BY 30 FINDS THE PATH FOR EACH OD PAIR THAT HAS THE
             C    HIGHEST COST BUT ALSO WITH POSITIVE PROPORTIONAL FLOW.
0126              DO 30 I=1,P
0127              DMX=0.0
0128              N3=M(I)+1
0129              N4=M(I+1)
0130              DO 31 J=N3,N4
```

```
0131              IF(Z(NO,J).EQ.0) GO TO 31
0132              DMX=DMAX1(DMX,C(J))
0133         31   CONTINUE
0134              J=M(I+1)
0135         32   IF(Z(NO,J).EQ.0) GO TO 35
0136              IF(DMX.EQ.C(J)) GO TO 33
0137         35   J=J-1
0138              GO TO 32
0139         33   L(I)=J
0140         30   CONTINUE
              C   IF IFLAG IS EQUAL TO 1 WE ARE ABOUT TO RESTART THE ALGORITHM OR STOP ALL
              C   TOGETHER IF IFLOG IS ALSO EQUAL TO 1
0141              GO TO 48
              C   WE GO TO 48 TO COMPUTE THE INTEGER LABELLING.
              C   ********************************************************************
              C   WE GO TO 42 WHEN WE WANT TO COMPUTE AN ARTIFICIAL VECTOR LABELLING.
0142         42   NUMBE=NUMBE+1
0143              DO 44 I=1,P
0144              IMX=0
0145              N3=M(I)+1
0146              N4=M(I+1)
0147              IF(N4-N3.EQ.0) GO TO 49
0148              DO 45 J=N3,N4
0149         45   IMX=MAX0(IMX,Z(NO,J)-ZOR(J))
0150              J=M(I)+1
0151         46   IF(IMX.EQ.Z(NO,J)-ZOR(J)) GO TO 47
0152              J=J+1
0153              GO TO 46
0154         49   J=M(I)+1
0155         47   L(I)=J
0156         44   CONTINUE
              C   ********************************************************************
              C   HERE WE COMPUTE THE INTEGER LABELLING.
0157         48   I=1
0158         41   IF(L(I).NE.M(I+1)) GO TO 40
0159              IF(I.EQ.P) GO TO 40
0160              I=I+1
0161              GO TO 41
              C   ********************************************************************
              C   IL CONTAINS THE INTGER LABEL. LAB TELLS US WHICH ROW OF Z HAS THE SAME LA
              C   BEL. WE REDEFINE LABEL AND THEN DETERMINE THE NEW PIVOT ELEMENT.
0162         40   IL=L(I) - I + 1
0163              LAB=LABEL(IL)
0164              LABEL(IL)=NO
0165              NO=LAB
0166              IF(NO.EQ.1.OR.NO.EQ.ISN) GO TO 51
0167              DO 50 I=1,N1
0168         50   Z(NO,I)=Z(NO-1,I)+Z(NO+1,I)-Z(NO,I)
0169              GO TO 55
0170         51   IF(NO.EQ.1) GO TO 53
0171              DO 52 I=1,N1
0172         52   Z(ISN,I) = Z(ISN-1,I) +Z(1,I) - Z(ISN,I)
0173              GO TO 55
0174         53   DO 54 I=1,N1
0175         54   Z(1,I)=Z(ISN,I)+Z(2,I) - Z(1,I)
              C   WE ARE NOW COMPUTING THE NEW PIVOT ELEMENT.
```

```
            C     ******************************************************************************
0176        55    WXY=CUTIME(0)
0177              IF(WXY.GT.TIME) GO TO 1987
            C     EVALUATE THE FIRST COMPONENT OF THE EXTRA SIMPLEX COMPONENT OF THE NEW PIV
            C     OT. IF LESS THAN 0 WE RESTART. IF 0 WE PERFORM A TRUE VECTOR LABELLING.
            C     IF GREATER THAN 0 WE GO TO ARTIFICIAL LABELLING. THESE ARE 100,19,42 RESP.
0178              IF(Z(NC,M(P+1)+1)) 100,19,42
            C     ******************************************************************************
0179        1987  IFLOG=1
0180              GO TO 114
            C     ******************************************************************************
            C     WE NOW PRINT OUT RESTART INFORMATION.
0181        100   IREST=IREST+1
0182              WRITE(6,101) IREST
0183        101   FORMAT( ' RESTART NUMBER:',I3)
0184        114   NO=LABEL(IL)
0185              DOX=0.0D0
0186              DO 70 I=1,P
0187              N3=M(I)+1
0188              N4=M(I+1)
0189              DO 71 J=N3,N4
0190              IF(Z(NO,J).EQ.0) GO TO 71
0191              DOX=DMAX1(DOX,C(L(I))-C(J))
0192        71    CONTINUE
0193              WRITE(6,72) I,(PAT(IK),IK=N3,N4)
0194        72    FORMAT( ' OD PAIR',I3,'FLOWS:',8F14.8/9F14.8/9F14.8)
0195              WRITE(6,74) (C(IK),IK=N3,N4)
0196        74    FORMAT( ' PATHCOSTS:',8F14.8/9F14.8/9F14.8)
0197        70    CONTINUE
0198              WRITE(6,108) DOX
0199        108   FORMAT( ' MAXIMUM DIFFERENCE OVER OD PAIRS OF PATHCOST MAXIMUM DIF
                 1FERENCE:',F16.8)
0200              OF=0.0D0
0201              DNX=0.0D0
0202              DO 112 I=1,Q
0203              WRITE(6,111) I,F(I),FCOST(I)
0204        111   FORMAT( ' LINK',I3,1X,'HAS FLOW',F12.6,1X,'AT COST',F16.6)
0205              OF=OF+OFC(I,F(I))
0206              DNX=DNX+FCOST(I)*F(I)
0207        112   CONTINUE
0208              WRITE(6,110) NUMBE,NUMEEE
0209        110   FORMAT( ' ARTIFICIAL EVA.',I10,'FUNC.EVA.',I10)
0210              WRITE(6,117) DNX
0211        117   FORMAT( ' TOTAL COST IF PLAN IS IMPLEMENTED:',F16.8)
0212              WRITE(6,118) OF
0213        118   FORMAT( ' NONLINEAR PROGRAMMING OBJ.FUNC.',F16.8)
            C     IF IFLOG IS EQUAL TO 1 THEN WE ARE ABOUT TO TERMINATE DUE TO TIME LIMIT
0214              IF(IFLOG.EQ.1) GO TO 129
            C     ******************************************************************************
            C     COMPUTE THE NEXT APPROXIMATION TO START THE ALGORITHM.
0215              IDM=0
0216              NO=LAB
0217              DO 103 I=1,N2
0218              ZOF(I)=0
0219              DO 102 J=1,ISN
0220        102   ZOF(J) = ZOF(J) + Z(J,I)
```

```
0221                ZOR(I) = ZOR(I) - Z(NC,I)
0222          103   CONTINUE
0223                DO 104 I=1,P
0224                N3=M(I)+1
0225                N4=M(I+1)
0226                ID(I)=0
0227                DO 105 J=N3,N4
0228                ZOR(J)= ZOR(J)*DINCR/(ISN-1)
0229          105   ID(I)=ID(I)+ZOR(J)
0230                ID(I)=ID(I)+1
0231          104   IDM=MAXO(IDM,ID(I))
0232                WXY=CUTIME(0)
0233          119   FORMAT( ' TIME UP TO THIS RESTART:',F14.8,'MAXIMUM SUM OF Z',I12)
0234                WRITE(6,119) WXY,IDM
0235                WRITE(6,2001)
0236          2001  FORMAT( ' ************************************************
             1*****************************************')
              C     IDM MAKES SURE WE DON'T HAVE INTEGERS APPROACHING 2 BILLION
0237                IF(IDM.LE.0) GO TO 121
0238                IF(DOX.IT.CLOSE) GO TO 125
0239                GO TO 18
0240          121   WRITE(6,122)
0241          122   FORMAT( ' INTEGER VARIABLE BIGGER THAN 2 BILLION')
0242                GO TO 129
0243          125   WRITE(6,126)
0244          126   FORMAT( ' PATHCOSTS ARE VERY CLOSE')
0245          129   WXY=CUTIME(C)
0246                WRITE(6,130) WXY
0247          130   FORMAT( ' CUMULATIVE TIME:',F16.8)
0248                STOP
0249                END
```

```
0001              REAL FUNCTION COST*8(I,J)
0002              INTEGER COEFF(55,4)
0003              REAL*8 A,B,C,D,J,DFLOAT
0004              COMMON/PRICE/COEFF
                C     PUT YOUR COST FUNCTION INBETWEEN THESE COMMENT CARDS
0005              COST=(COEFF(I,3)*J +COEFF(I,2))*J + COEFF(I,1)
                C     ******************************************************************************
0006            3     RETURN
0007              END
```

```
FORTRAN IV G LEVEL  21                  OFC              DATE = 77153        19/49/52            PAGE 0001

0001                REAL FUNCTION OFC*8(I,J)
0002                INTEGER COEFF(55,4)
0003                REAL*8 A,B,C,D,J,DFLOAT
0004                REAL*8 ILOG
0005                COMMON/PRICE/COEFF
          C         PUT YOUR NONLIN. PRO. OBJ. FUNC. INBETWEEN THESE COMMENT CARDS
0006                OFC=((COEFF(I,3)*J/3.0D0+COEFF(I,2)/2.0D0)*J + COEFF(I,1))*J
          C         ********************************************************************************
0007      3         RETURN
0008                END
```

5. Solved Test Problems

To test the effectiveness of PATHFIX, we have solved three test problems and in this section we compare the solutions obtained versus established solution methods. The three problems were (a) the four node, five link sample problem of Section 3; (b) thethree node, five link problem of Potts and Oliver[1], and (c) the nine node, thirty-six link problem of Steenbrink.[2] All computer runs were made on an IBM 360/91. The Frank-Wolfe algorithm employed was embodied in the TRAFFIC code developed by Florian and Nguyen at the University of Montreal.

The tables 5.1, 5.2, and 5.3 summarize the comparisons made. It is evident that the primary advantage of PATHFIX is its accuracy. On all problems the solutions are more precise than the alternative methods. It is particularly interesting that the PATHFIX algorithm yields a far more accurate solution than the often used Frank-Wolfe algorithm, even after 50 iterations of the latter, on small problems. This advantage must, of course, be weighed against the requirement that paths between OD pairs are a required input.

From these results it seems reasonable to conclude that PATHFIX would be of greatest use in (a) a restriction algorithm which selects paths for potential positive flows, (b) small assignment problems such as those arising in traffic control models, or (c) in aggregated problems with few paths.

---

1 Potts and Oliver. Flows in Transportation Networks. Academic Press, 1972, page 96.

2 Steenbrink, P.A., Optimization of Transport Networks, John Wiley, 1974, page 108.

Four Node Problem

	PATHFIX SOLUTION	FRANK-WOLFE SOLUTION (50 iterations)	EXACT SOLUTION
Link 1	2.000000	1.9392	2
2	0.000000	0.0608	0
3	1.000000	1.0528	1
4	1.000000	0.9472	1
5	3.000000	2.9920	3
Obj. Value	37.499998	37.560667	37.5
CPU Seconds	0.29	2.5	N/A

Table 5.1

Potts and Oliver Problem

	PATHFIX SOLUTION	FRANK-WOLFE SOLUTION		POTTS AND OLIVER SOLUTION
		(50 IT.)	(215 IT.)	
Link 1	4.642363	4.7988	4.651	4.642
2	1.326307	1.2786	1.342	1.326
3	4.673693	4.7919	4.691	4.674
4	2.316056	2.5909	2.342	2.316
5	4.683944	4.4738	4.690	4.684
Obj. Value	9.058692	9.6805	9.169	9.059
CPU Seconds	7.48	2.8	12.1	N/A

Table 5.2

Steenbrink Problem

		PATHFIX SOLUTION	STEENBRINK QUADRATIC PROGRAMMING SOLUTION
LINK	12	37.04	39
	13	592.59	562
	21	592.59	562
	24	1698.11	1694
	27	1558.35	1582
	28	1150.94	1162
	31	37.04	39
	35	1461.54	1430
	37	701.42	720
	38	1000.00	1011
	42	276.36	236
	46	283.02	275
	48	100.00	100
	49	716.98	725
	53	188.68	199
	58	34.96	65
	59	176.36	136
	65	283.02	275
	72	162.96	161
	73	1407.41	1438
	78	689.40	703
	82	23.64	64
	83	111.32	101
	84	943.34	918
	85	1192.31	1206
	89	704.64	752
	94	534.85	524
	95	1063.13	1089
	(Links with zero flows omitted)		
Obj. Value:		16961	16970
C.P.U. Seconds:		18.84	N/A

Table 5.3

APPENDIX
REPORT OF INVENTIONS

Although no inventions were developed in the course of this research, several improvements to transportation planning methodology were made. In Volume One, Section Four, a method for bounding errors in traffic assignment problems is presented. In Volume Two an entirely new approach to solving traffic assignment problems is presented. An experimental computer program of this new algorithm is discussed, and a listing of that program appears in Section Two.

110 Copies

U. S. DEPARTMENT OF TRANSPORTATION
TRANSPORTATION SYSTEMS CENTER
KENDALL SQUARE, CAMBRIDGE, MA. 02142

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE, $300

POSTAGE AND FEES PAID
U. S. DEPARTMENT OF TRANSPORTATION
518