OPTIMAL ROUTING OF URBAN TRAFFIC

by

ALAN MARTIN HERSHDORFER
A.B., Princeton University (1958)
M.S., Massachusetts Institute of Technology (1960)

Submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

Massachusetts Institute of Technology

June, 1965

Signature of Author ........................................

Department of City and Regional Planning, May 14, 1965

Certified by ........................................ Thesis Supervisor

Accepted by ........................................ Chairman, Departmental Committee on Graduate Students
ABSTRACT

OPTIMAL ROUTING OF URBAN TRAFFIC

by

ALAN MARTIN HERSHDORFER

Submitted to the Department of City and Regional Planning on May 14, 1965 in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

The design of an adequate road transportation network plays a major role in comprehensive urban and regional planning. With conventional road planning techniques, improvements may be proposed for local sections of a road network, without adequate evaluation of the effects of these improvements on the performance of the road system as a whole. Information on overall effects of network changes can lead to an improvement schedule significantly different from that developed according to purely local considerations.

A mathematical model based on linear programming techniques has been devised to measure the total effects of local system changes. Input to the model consists of: the connectivity of the road network, the description of each road link by a volume-velocity relationship, and the required origin-destination volumes. The solution determines an optimal routing pattern in the sense of minimum total travel time. This solution places a bound on the performance which may be expected from the network and forms a point of reference for the evaluation of flow patterns generated by other traffic assignment techniques, such as the sequential minimum path method.

If the model were applied to evaluate the effect of any single decision regarding changes in the network, one would be faced with prohibitive computational costs because the number of alternative decisions is large. This thesis presents a new formulation of the linear programming model, which is designed to determine in a computationally efficient manner not only optimal flows, but also optimal improvements to the existing network. To achieve this, we have extended the linear program to a mixed-integer programming formulation in a manner which allows the intervention of human decisions on design alternatives via a direct-access time-shared computer facility.

The mixed-integer formulation is applied to the following problems: 1) What is the optimal flow pattern and travel time cost given a network and set of transfer volumes? 2) The cheapest and quickest improvement is to be obtained from set-
ting streets one-way. This has the effects of adding lanes to the heavily travelled flow direction, but also of diverting traffic. Taking both these effects into account, which streets should be made one-way? 3) If the optimal performance obtainable by properly orienting existing road capacity is unsatisfactory, it is necessary to construct additional road capacity. Which road links should be constructed or improved so that total social cost (road user cost plus construction cost) is minimized, or so that road user cost drops to a specified level and construction cost is minimized?

These problems were investigated for sample networks under a range of traffic flow intensities and patterns. The following conclusions were drawn from the computations:

1) The Land and Doig algorithm applied to network synthesis problems has the property that a near optimal solution to the mixed-integer program can be obtained very rapidly. Determination of the optimal solution may take considerably more computation. However, when the flow pattern on the network shows strong directionality, the mixed-integer program converges rapidly to the optimal solution. The algorithm is an efficient tool for network planning.

2) In general the optimal solution could not be determined by inspection. However, examination of optimal flow patterns showed that one-way street configurations which cause extensive diversion of traffic have adverse effects on total travel time and the ability of the network to transmit flow. This suggests a conservative, heuristic approach which does not require a computer. That is, the most likely candidates for one-way streets are those with negligible flow in one direction. Such an approach minimizes rerouting but may not yield a great improvement. Complete execution of the mixed-integer algorithm will reveal bolder solutions, when they exist.

The decomposition algorithm of linear programming is applied to compare the route selection procedure in minimal path assignments with that used in system optimal assignments.

Thesis supervisor: Aaron Fleisher
Title: Associate Professor of City and Regional Planning
ACKNOWLEDGEMENTS

This research has received support from a number of sponsors. The first year of thesis research was supported by the Title IV Fellowship Program of the National Defense Education Act. Subsequent research support came from the S. A. Stouffer Research Fellowship of the Harvard - M.I.T. Joint Center for Urban Studies, and the Joint Highway Research Project of the Massachusetts Department of Public Works and the M.I.T. Department of Civil Engineering. Computer facilities were contributed by the M.I.T. Project MAC Time-Shared Computer Facility.

The author would like to thank his advisor, Professor Aaron Fleisher, for many stimulating discussions. The author is particularly indebted to Professor C. L. Miller, head of the M.I.T. Department of Civil Engineering for permitting him to devote his complete attention to this research project.

We appreciate the very efficient and accurate typing by Miss Christina Liljencrantz. Mr. Harold C. Frazier was very helpful in the preparation of the computer program.

Finally the author wishes to express his appreciation to his wife, Selma A. Hershdorfer, for designing and executing the illustrations to the text.
TABLE OF CONTENTS

Title page 1
Abstract 2
Acknowledgments 4
Table of Contents 5
List of Figures 9

CHAPTER I Economic Location Theory and Network Flow Traffic Models 13
1.1 Introduction 13
1.2 Transportation as an Economic Good 14
1.3 The Economic Role of Transportation Systems - Theoretical Aspects 15
   1.3.1 Prediction of traffic flow patterns - demand and equilibrium 15
   1.3.2 Efficient utilization of the road system 17
   1.3.3 Transportation costs and residential location theory 20
1.4 Constructive Solutions to the Traffic Prediction Problem 23
   1.4.1 Linear programming and game theoretical models of traffic flow patterns 23
   1.4.2 Game theoretical models and metropolitan traffic planning techniques 25
1.5 System Effects of Network Modification 29
   1.5.1 The traffic assignment technique as an evaluative device 29
   1.5.2 Combinatorial problems and mathematical programming 31
1.6 Summary 33
1.7 Notes to Chapter I 35

CHAPTER II Development of a Mixed-Integer Mathematical Programming Formulation for Determining Optimal Improvements to a Traffic Network 37
2.1 Introduction 37
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2 Physical Statement of the Basic Problem</td>
<td>39</td>
</tr>
<tr>
<td>2.2.1 The general physical problem</td>
<td>39</td>
</tr>
<tr>
<td>2.2.2 The particular form of the physical problem</td>
<td>40</td>
</tr>
<tr>
<td>2.3 Parameters Required to Specify the Physical Problem</td>
<td>49</td>
</tr>
<tr>
<td>2.4 Transition from Empirical Data to a Theoretical Representation:</td>
<td>50</td>
</tr>
<tr>
<td>Analysis of the Link Average Travel Time Function and the Link Capacity Limit</td>
<td></td>
</tr>
<tr>
<td>2.4.1 Introduction</td>
<td>50</td>
</tr>
<tr>
<td>2.4.2 The empirical link average travel time function</td>
<td>51</td>
</tr>
<tr>
<td>2.4.3 Reinterpretation of the empirical link average travel time function</td>
<td>54</td>
</tr>
<tr>
<td>2.4.4 Flow density and flow rate</td>
<td>59</td>
</tr>
<tr>
<td>2.4.5 Relating density, flow rate, and average travel time</td>
<td>63</td>
</tr>
<tr>
<td>2.5 From Average to Total and Incremental Travel Time Functions</td>
<td>74</td>
</tr>
<tr>
<td>2.5.1 The empirical total travel time function for a link</td>
<td>75</td>
</tr>
<tr>
<td>2.5.2 The approximate total travel time function for a link and the arc-pair concept</td>
<td>77</td>
</tr>
<tr>
<td>2.5.3 Additional properties of the arc-pair concept</td>
<td>81</td>
</tr>
<tr>
<td>2.5.4 Algebraic form of the travel time functions</td>
<td>86</td>
</tr>
<tr>
<td>2.6 Matrix Representation of the Problem: The Network Configuration and Commodity Input-Output Data</td>
<td>96</td>
</tr>
<tr>
<td>2.6.1 The node-arc incidence matrix</td>
<td>96</td>
</tr>
<tr>
<td>2.6.2 Conservation of flow equations</td>
<td>96</td>
</tr>
<tr>
<td>2.6.3 Capacity constraints</td>
<td>104</td>
</tr>
<tr>
<td>2.6.4 Average arc travel time</td>
<td>106</td>
</tr>
</tbody>
</table>
2.7 Extension of the Model to Problems of Optimal Network Design

2.7.1 The variable capacity formulation
2.7.2 The one-way street model
2.7.3 The optimal link addition problem
2.7.4 The optimal cost improvement problem

2.8 Notes to Chapter II

CHAPTER III Computational Experience with Mixed-Integer Mathematical Programming and Network Models

3.1 Introduction
3.2 Solution of Mixed-Integer Programs

3.2.1 Lattice points in the feasible region
3.2.2 Searching among lattice points
3.2.3 A two-dimensional example
3.2.4 Constructing the feasible domain of an integer variable as a function of the objective value
3.2.5 Obtaining an ordering rule by considering the convexity of the integer feasibility diagram
3.2.6 The general branch and bound algorithm

3.3 Computational Implementation
3.4 The One-Way Street Problem with Three Commodities

3.4.1 Loading levels and loading patterns
3.4.2 The isotropic pattern
3.4.3 The biased loading pattern
3.4.4 The lopsided loading pattern
3.4.5 The one-way street problem: conclusions

3.5 The Link Addition Problem

3.5.1 An example and its solution
3.6 Summary
3.7 Notes to Chapter III
<table>
<thead>
<tr>
<th>CHAPTER IV</th>
<th>Strategies, Decomposition, and Priorities in the Multi-Commodity Flow Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>4.2</td>
<td>The Routing Plan Example</td>
</tr>
<tr>
<td>4.3</td>
<td>Game Theoretical Interpretations</td>
</tr>
<tr>
<td>4.4</td>
<td>Decomposition and the Multi-Commodity Flow Problem</td>
</tr>
<tr>
<td>4.5</td>
<td>Simplex Multipliers in the Multi-Commodity Flow Problem</td>
</tr>
<tr>
<td>4.6</td>
<td>Summary</td>
</tr>
<tr>
<td>4.7</td>
<td>Priorities on the Network</td>
</tr>
<tr>
<td>4.8</td>
<td>Notes to Chapter IV</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHAPTER V</th>
<th>Conclusions and Recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Conclusions</td>
</tr>
<tr>
<td>5.2</td>
<td>Recommendations</td>
</tr>
<tr>
<td>5.3</td>
<td>Notes to Chapter V</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Appendix A</th>
<th>A Survey of Research on the Multi-Commodity Flow Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1</td>
<td>The Charnes-Cooper Multi-Copy Traffic Network Model</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Appendix A</th>
<th>Problems Related to the Multi-Copy Traffic Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.2</td>
<td>Multi-Commodity Feasibility</td>
</tr>
<tr>
<td>A.2.1</td>
<td>The solid transportation problem</td>
</tr>
<tr>
<td>A.2.2</td>
<td>The arc-chain formulation for multi-commodity flows</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Appendix B</th>
<th>Bibliography</th>
</tr>
</thead>
</table>

Biography
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Section</th>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.2</td>
<td>A</td>
<td>Network of Arterial Roads Lying Between the Central Business District and Freeway Entrance</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Characterization A of Rush Hour Flow Pattern</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Characterization B of Rush Hour Flow Pattern</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>Network of Arterial Roads Showing Node and Link Numbers</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>Desire Line Pattern Corresponding to Characterization B</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>Network of Inputs and Outputs Corresponding to Characterization B</td>
<td>47</td>
</tr>
<tr>
<td>2.4.2</td>
<td>A</td>
<td>Capacity Function, Interpretation 1</td>
<td>53</td>
</tr>
<tr>
<td>2.4.3</td>
<td>A</td>
<td>Volume–Time Relationships for Typical Road Sections</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Capacity Function, Interpretation 2</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Capacity Function, Interpretation 3</td>
<td>58</td>
</tr>
<tr>
<td>2.4.4</td>
<td>A</td>
<td>Fundamental Diagram of Traffic Flow</td>
<td>62</td>
</tr>
<tr>
<td>2.4.5</td>
<td>A</td>
<td>Construction to Illustrate Derivation of ( f(d) ) from ( t(f) )</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Reflected Fundamental Diagram</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Density ( d(f) ) Based on ( t(f) ) Interpretation 1</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>Reflected Fundamental Diagram, Based on ( t(f) ) Interpretation 1</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>Reflected Fundamental Diagram Generating ( t(f) ) Interpretation 4</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>Capacity Function, Interpretation 4</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>Density ( d(f) ) Based on ( t(f) ) Interpretation 4</td>
<td>73</td>
</tr>
<tr>
<td>Section</td>
<td>Figure</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>2.5.1</td>
<td>A</td>
<td>The Empirical Total Travel Time Function for a Link</td>
<td>76</td>
</tr>
<tr>
<td>2.5.3</td>
<td>A</td>
<td>Incremental Travel Time Function and the Arc Pair</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Incremental Travel Time: ( f )</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Incremental Travel Time: ( f_c )</td>
<td>85</td>
</tr>
<tr>
<td>2.5.4</td>
<td>A</td>
<td>Comparison of Empirical and Derived Average Travel Time Functions</td>
<td>89</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Derived Average Travel Time Function</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Comparison of Incremental Travel Time Function for Link and Arcs</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>Comparison of Total Travel Time Function: for Link and Arcs</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>Comparison of Average Travel Time Function for Link and Arcs</td>
<td>95</td>
</tr>
<tr>
<td>2.6.1</td>
<td>A</td>
<td>Example Network to Illustrate Node-Arc Incidence Matrix</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Node-Arc Incidence Matrix for Problem Network</td>
<td>98</td>
</tr>
<tr>
<td>2.6.2</td>
<td>A</td>
<td>Illustration of Commodity Input-Output Requirements</td>
<td>102</td>
</tr>
<tr>
<td>2.7.1</td>
<td>A</td>
<td>Fractional Switching: Incremental Time Function Shift</td>
<td>115</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Fractional Switching: Total Time Function Shift</td>
<td>116</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Fractional Switching: Average Travel Time Function Shift</td>
<td>117</td>
</tr>
<tr>
<td>3.2.1</td>
<td>A</td>
<td>Polygon and Included Lattice Points</td>
<td>143</td>
</tr>
<tr>
<td>3.2.3</td>
<td>A</td>
<td>Integer Programming Example</td>
<td>147</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Integer Programming Example</td>
<td>149</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Integer Programming Example</td>
<td>150</td>
</tr>
<tr>
<td>Section</td>
<td>Figure</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.2.4</td>
<td>A</td>
<td>Feasible Ranges of $d_1$ and $d_2$ for $T = T_n$</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Falling objective Plane and Feasible Ranges</td>
<td>153</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Integer Feasibility Diagram</td>
<td>155</td>
</tr>
<tr>
<td>3.2.5</td>
<td>A</td>
<td>Tree Diagram for Example Problem</td>
<td>158</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Classification of Transitions</td>
<td>163</td>
</tr>
<tr>
<td>3.2.6</td>
<td>A</td>
<td>Generalized Branch and Bound Tree</td>
<td>165</td>
</tr>
<tr>
<td>3.4.2</td>
<td>A</td>
<td>Suggested Network Orientation for Isotropic Loading</td>
<td>184</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Total Cost as a Function of Total Load for Three Network Configurations</td>
<td>186</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Isotropic Load, Level 950, Flows on Symmetric Network</td>
<td>189</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>Isotropic Load, Level 950, Flows on Free Network</td>
<td>190</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>Isotropic Load, Level 950, Free Network Orientation</td>
<td>192</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>Isotropic Load, Integer Program</td>
<td>194</td>
</tr>
<tr>
<td>3.4.3</td>
<td>A</td>
<td>Biased Load, Integer Program</td>
<td>197</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Biased Load, Integer Solution A</td>
<td>199</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Biased Load, Integer Solution B</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>Biased Load, Integer Solution C</td>
<td>201</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>Biased Load, Integer Solution D</td>
<td>202</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>Biased Load, Free Optimal Flow Pattern</td>
<td>205</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>Biased Load, Free Network, Optimal Decision Variables</td>
<td>206</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>Biased Load, Orientation Resulting from Rounding Free Decision Variables</td>
<td>207</td>
</tr>
<tr>
<td>3.4.4</td>
<td>A</td>
<td>Lopsided Loading, Flows on Symmetric Network</td>
<td>209</td>
</tr>
<tr>
<td>Section</td>
<td>Figure</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.4.4</td>
<td>B</td>
<td>Lopsided Loading, Flows on Free Network</td>
<td>210</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Lopsided Loading, Optimal Free Configuration</td>
<td>211</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>Lopsided Loading, Optimal Free Configuration Rounded</td>
<td>212</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>Lopsided Loading, Optimal Integer Configuration</td>
<td>213</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>Lopsided Loading, Flows on Optimal Integer Configuration</td>
<td>214</td>
</tr>
<tr>
<td>3.5.1</td>
<td>A</td>
<td>Link Addition Example</td>
<td>218</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Link Addition Integer Program</td>
<td>219</td>
</tr>
<tr>
<td>4.2</td>
<td>A</td>
<td>Example Network</td>
<td>230</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Basic Solutions for Commodity M</td>
<td>232</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Basic Solutions for Commodity N</td>
<td>233</td>
</tr>
</tbody>
</table>
CHAPTER I
Economic Location Theory and Network
Flow Traffic Models

1.1 Introduction

The main content of this thesis is the development of a mathematical technique for designing or modifying a road network such that an aggregate least cost utilization of the network can be effected. The development of the mathematical model basic to this technique is the subject matter of Chapter II. Essential to this model is the notion of an aggregate minimal cost flow pattern. This pattern may be contrasted with the flow pattern which results when the driver of each vehicle on the network seeks to minimize his own trip length. Calculation of the latter type of pattern has served in the evaluation of present and projected transportation networks in numerous metropolitan planning projects throughout the United States during the past decade.

That formulation does not lend itself to the determination of network improvements except through the enumeration and evaluation of possible alternative designs. On the other hand, we shall show that a model based on aggregate minimum cost flow patterns can be used to determine directly the best network improvement policy. This has the great advantage that large numbers of alternatives need not be evaluated; Furthermore, the policy which is chosen is known to be the best. In an enumerative scheme, the possibility of an untried alternative being a better solution than the one selected remains, unless the evaluation of alternatives is exhaustive. Usually the computational cost of an exhaustive search is prohibitive.
The minimum aggregate (or system) cost flow pattern — minimum path flow pattern dichotomy appears in the literature of modern economic location theory, as well as in the literature of metropolitan transportation planning.

In this introductory chapter, we should like to trace the development of transportation models based either on minimum path flow patterns or on minimum system cost flow patterns.

The conclusions drawn from this historical survey will motivate our formulation of a new model for the synthesis of road traffic networks.

1.2 Transportation as an Economic Good

Economic location theory, an extension of classical rent theory, studies the effect of the competition for productive space on the distribution of land uses. William Alonso, in *Location and Land Use*, gives a clear and succinct review of location theory (1). Alonso demonstrates the dependence of locational theory on Ricardo's fundamental notion of rent as a measure of the differences in productivity between units of agricultural land. Von Thünen explicitly included savings in transportation costs in the rent calculation: "The rent at any location is equal to the value of its product minus production costs and transport costs" (2).

R. M. Haig stressed the complementarity of rent and transport costs: "Transportation is a device to overcome the 'friction of space' ... the user of a site pays as the 'costs of friction' transport costs and rent, which is 'the saving in transport costs' ... . 'The theoretically perfect site for an activity is that which furnishes the desired degree of accessibility at the lowest costs of friction.'"
Alonso points out that Haig views the sum of rent and transport costs as varying with the productivity of a site (in terms of volume of retail sales, for example). This variation implies that a principle of total minimizing of "costs of friction" does not suffice to determine the pattern of land uses, unless "revenue and all other costs are held constant". (4). The assumption of constant revenue is artificial; retail sales depend on accessibility and the presence or absence of competition. Haig's theory has been criticized on a number of other grounds, including its neglect of site size and its failure to consider social factors in addition to "costs of friction" as determinants of residential location patterns. Haig's theory is significant from our viewpoint because it emphasis on the complementarity of rent and transport costs has strongly influenced the thinking of Beckman and Wingo.

1.3 The Economic Role of the Transportation System

1.3.1 Prediction of traffic flow patterns - demand and equilibrium

*Studies in the Economics of Transportation*, a Cowles Foundation Report by Martin Beckman, C. B. McGuire, and Christopher B. Winsten (5) has had an immense effect on American metropolitan planning in the last decade, for reasons we shall presently discuss. From the point of view of location theory, this book contains notions basic to Beckman's and Wingo's subsequent models of residential location. In terms of metropolitan planning in general, the ideas in
this book have appeared as the framework for the development of models for traffic prediction and land use development which have been implemented by large-scale metropolitan traffic planning projects throughout the United States (and Toronto, Canada), although Beckman's work is not acknowledged in metropolitan traffic planning literature.

We are concerned with Part I of the report by Beckman et al, which discusses vehicular traffic. Chapter I summarizes the known experimental data regarding the costs of congestion to road traffic: Travel cost on a route is an increasing function of flow on the route. Chapter II analyses the demand for transportation between two points as a decreasing function of travel cost between the two points. The shortest route between two points is emphasized as, the route of minimum travel cost in units of time per unit of flow. It is explicitly assumed that traffic will tend to move along minimum time paths.

This assumption is a restatement of "Wardrop's Second Principle" (6): In any set of flows on a network only those paths having equal travel time (i.e., equal to the shortest path) between origin and destination, will be used. Beckman's further argument relies strongly on this assumption; Beckman acknowledges Wardrop's influence.

In Chapter III the question of the existence of equilibrium flow on the network is considered.

The problem is stated as follows: the demand for travel between any two points on a network is a decreasing function of flow on that path. Any demanded flow will be assumed to follow the shortest route on routes between those points. If the flow pattern is repeated periodically, information as to
route costs will accumulate. This will lead to changes of route and possibly of demand. Question: does the system settle down to an equilibrium state of demands and route choices? Beckman derives the following conclusion: "An equilibrium always exists if demand is a decreasing function to trip cost and transportation cost is a constant or increasing function of traffic flow, and that equilibrium is unique whenever the shortest routes between all pairs of locations are unique and cost is strictly increasing with increasing flow". (7).

Beckman goes on to show that when a non-unique equilibrium is not stable, one may obtain a perpetual cycling among various choices of routes.

1.3.2 Efficient utilization of the road system

In Chapter IV, Beckman considers the problem of efficient usage of the road system in the following terms:

"In the case of two roads between a pair of locations, traffic distributes itself so that the average transportation cost becomes the same on either road. If one road is shorter but of small capacity, the delays at equilibrium due to the more crowded conditions on the shorter road would just compensate for the greater operating costs on the longer road. Congestion on the short road, by discouraging further traffic there has led to a diversion of some traffic to the long road ... .

"In the case of the equilibrium distribution, as Pigou has pointed out, it would be possible, by shifting a few cars from one road to the other, greatly to lessen the trouble of driving for those left on the congested road, while only slightly increasing the trouble of driving on the less congested road."
"In these circumstances a rightly chosen measure of differential taxation against (the congested road) would create an artificial situation superior to the natural one. But the measure of differentiation must be rightly chosen."

In the remainder of the chapter, this theme is elaborated. It is argued that the road users' time, money costs, and risk are increasing functions of total flow on a link; time is accepted as the major component of user cost, and the addition of a risk value and money cost do not greatly change the shape of the cost function.

The problem of the two roads between two points with fixed total flow is analyzed graphically. It is shown that the equilibrium solution (with equal average costs on both roads) is not the solution which minimizes total cost. The reason for this lies in the nature of the traffic congestion phenomenon.

"From a certain flow on, each additional vehicle causes some delay and risk to others present for which it does not bear the cost. As Professor Frank Knight has put it for the case of two alternate highways,

'The congestion and interference resulting from the addition of any particular truck to the stream of traffic on the narrow but good road affects in the same way the cost and output of all the trucks using that road. It is evident that if, after equilibrium is established, a few trucks should be arbitrarily transferred to the broad road, the reduction in cost or increase in output to the trucks remaining on the narrow road would be a clear gain to the traffic as a whole' " (9).
Beckman terms this difference the "excess of social incremental cost over private incremental cost". "If the road user were to bear his full share of the cost, he would pay a tax or toll equal to this excess. This would make him realize the cost he causes to others and thus provide an incentive to keep social cost down by making the proper choices. This is indeed Pigou's 'rightly chosen measure of differential taxation'" (9).

Beckman suggests that this differential taxation be applied in the form of "efficiency tolls" on roads. Admitting that the calculation of these tolls would present a difficult problem for a non-trivial network, Beckman suggests that the theory of linear or convex programming can provide the solution to the problem of finding the most efficient choice of routes when demands are fixed, and observes that the solution to the dual linear program (the "simplex multipliers") can be interpreted as trip costs. We shall show in a later chapter that the simplex multipliers in fact provide the solution to the problem of calculating the efficiency tolls. We shall apply the decomposition algorithm of linear programming to demonstrate the solution to Beckman's two route problem in such a way that the meaning of the multipliers as tolls will become evident, and the method of solution will moreover be directly extendable to networks arbitrary size!

We conclude, according to Beckman, that traffic congestion for travel plays a crucial part in the determination of demands for travel between points, and on the routes chosen between those points. Beckman pointed out that an equilibrium solution can be one in which the average cost is the same over all routes used between two points, but that
this solution need be neither stable nor one which uses the public facilities most efficiently.

We stated above that the work of Beckman, Winsten, and McGuire has had implications for location theory and for transportation planning. We next discuss its impact on residential location theory.

1.3.3 Transportation costs and residential location theory

In the model discussed above, Beckman sought an equilibrium solution to the problem of unknown demands for travel on a network, and unknown choices of routes arising from any particular pattern of demand. Beckman subsequently (1957) treated the distribution of residential density about a central city. Assuming the complementarity or rent and transportation cost, a housing plus travel cost consumption function, and a linear commuting cost function, Beckman determined a market solution of rent and residential densities in which the higher income families reside on the periphery of the city. Each family was assumed to choose its location (radius) so as to maximize site size given budgetary constraints (10). Here Beckman has implicitly solved a special case of the equilibrium problem discussed above. He assumed all destinations to be at the center of the city.

The average family is assumed to allocate a fixed portion of its income to the sum of travel costs plus land costs in such a way that the family maximizes the living space it can obtain for its housing expenditure. These assumptions plus the geometrical symmetries of the problem determine the solution. However, the linear commuting cost function avoids the problem of congestion costs.
Following directly in the footsteps of Beckman, Wingo (11) further studied the distribution of residences, placing more emphasis on congestion in transportation costs. He introduced the notion of "ingression loss" as follows:

The transportation network into the city has a limited capacity. Any route can transmit traffic at a rate bounded by its capacity. As there is a positive value attached to commuting time, travelers will compete for the most efficient channels. But those who first enter the channels effectively "seize the capacity"; they force others to accept less efficient facilities. This means that the others must take longer routes, or enter shorter routes later. If a commuter wishes not to be forced onto an inferior facility, he must leave earlier in order to seize the desirable facility for himself. The loss of time he incurs because he must leave early to compete for the limited capacity of the desirable facility is called the ingression loss. As in Beckman et al, Wingo emphasizes the difference between the marginal cost associated with the addition of one vehicle to the network, and the increase in travel cost averaged over all vehicles in the system when one vehicle is added. This difference is a congestion loss and hence a net disutility to the system. Wingo sees these disutilities as occurring from the finite capacity of the transportation system. If there were no ingression losses due to competition for limited capacity there would similarly be no congestion losses due to overloaded facilities. This suggests that system utility could be maximized (or social disutility minimized) in two ways:

(a) Optimal routing of flows on the given network

(b) Optimal improvements of the existing network (i.e., modifying the network so that total congestion losses
are reduced to the greatest extent possible for a given level of improvement cost).

The themes of optimal routing and optimal network improvement form the essential content of this thesis.

William Alonso, in Location and Land Use, acknowledges that his theory is an extension of Beckmans (12). Instead of resting with a strict complementarity of rent and transportation as determinants of locational preference, Alonso keeps preferences for land and accessibility apart from budgetary constraints, joining them in terms of marginal rates of substitution and marginal costs (13).

Alonso merges the Haig-Beckman emphasis on budgetary determinants of residential location, with the Park-Burgess-Hoyt-Firey insistence on the role of social tastes and preferences. If Beckman's ideas have appeared implicitly in many metropolitan transportation planning projects in the United States, (and in Toronto, Canada, especially), as we shall presently show, Alonso's theory has had a specific impact on the Penn-Jersey Study.

Beckman's major points were: (a) the existence of an equilibrium pattern of traffic demands and flows with increasing travel cost functions and decreasing demand functions: active routes are those with travel times equal to the minimum time path to the destinations, (b) the instability of the equilibrium, (c) the difference between a system optimum solution and that obtained by requiring that any active route between an origin and a destination be a minimum path route. Studies in the Economics of Transportation was first published in January 1956, and reprinted in February, 1957 and in January, 1959.
1.4 Constructive Solutions to the Traffic Prediction Problem

1.4.1 Linear programming and game theoretical models of traffic flow patterns

A. Charnes and W. W. Cooper published in 1958 (4) a linear programming formulation for simulating road traffic, with fixed demands and increasing travel cost functions. The work appears to be an application to road traffic of a linear programming formulation originally developed for the simulation of the behavior of firewater safety systems in refinery operations, which also forms a network flow problem with increasing cost functions (15). There Charnes and Cooper transform a convex program to a linear program by adding parallel arcs. In the following chapter we shall discuss this technique in detail. In "Extremal Principles for Simulating Traffic Flow", and again in December 1959 at the Symposium on the Theory of Traffic Flow held at Warren, Michigan (16), Charnes and Cooper applied this formulation to road traffic. The discussion of Charnes and Cooper bears a remarkable parallelism to the analysis of Beckman et al. Two aspects of the traffic simulation problem are discussed: the minimum cost (system optimum flow pattern) flow pattern for fixed demands and increasing travel costs, and the problem of equilibrium flows when the demands are fixed and minimum path routes are chosen. It would appear that Beckman et al, and Charnes and Cooper have all been influenced by Wardrop's (6) dichotomy of minimum time path patterns and minimum total time cost patterns.

Thus in Charnes and Cooper as in Beckman et al, system optimum flow patterns are contrasted with minimum path flow patterns. Charnes and Cooper do not refer to Beckman's work;
they treat the same problems in a rather different language. While Beckman uses linear and convex programming analysis for a theoretical discussion of the conditions determining an optimum flow pattern, Charnes-Cooper present the solution of an "actual" network example problem based on data supplied by the Chicago Area Transportation Study. It is interesting to note that the Chicago Area Transportation Study, one of the first in the nation, was exposed to the linear programming approach through contact with Charnes at an early date. Yet linear programming analysis has played little or no part in the transportation planning models developed on a large scale by metropolitan transportation studies in the United States. We shall show in this thesis that the linear programming method can make a valuable contribution to transportation planning.

On the other hand, the minimum path aspect as formulated by Charnes has been of fundamental significance for American metropolitan transportation studies. Charnes states the problem of determining an equilibrium set of network flows for fixed demands in the language of game theory. He associates with each origin a player who must choose a set of routes such that the required (demanded) number of vehicles will travel from the origin to each destination at the minimum total travel time for the vehicles associated with that origin. Due to the interaction of vehicles from different origins the travel times as seen by a given player depend on the actions of the other players. Charnes-Cooper remark that this suggests an iterative approach to the determination of the equilibrium set of flows on a network. Each player chooses his set of routes (chooses a strategy or routing plan) in turn. After all players have had a turn, the resultant travel times are revealed, and the players again take turns in revising
their routing strategies. Eventually the flows approach an equilibrium in which each player would worsen his position if he attempted to change his strategy. Charnes-Cooper refer to this as a Nash equilibrium and remark that a Nash equilibrium is required to be stable only against deviations by one player at a time. We shall later give a simple two person example of this game which is stable against single person deviations, and against two-person deviations unless there is cooperation among the players. In our example the stable situation will be a Pareto optimum: it will be impossible to change the situation without making at least one person worse off. But, this stable solution to the two person game will not be a system optimum solution. We shall also give an example of this game which is instable for two-player deviations, and cycles between two solutions.

Note that the Charnes-Cooper formulation is a restatement of the Beckman problem of equilibrium in the network (holding demands fixed, however). Charnes-Cooper proposed an iterative (constructive) solution technique where Beckman stated the conditions for the existence of an equilibrium. We conclude that Charnes-Cooper visualized a traffic distribution problem with much the same characteristics as the problem of Beckman et al, but carried the analyses of both the equilibrium question and the optimality question further. Moreover the Charnes-Cooper game theoretical model forms the bases of all metropolitan traffic assignment methods, in a manner which we shall now outline.

1.4.2 Game theoretical models and metropolitan traffic planning techniques

The Traffic Assignment Manual (17), which is the user's handbook for the Bureau of Public Roads Traffic Assignment
Program, begins with a brief history of computer programs which have been developed to determine the flow pattern of traffic on a road network for fixed demand. According to this source (18) the earliest computer program was developed "about" 1957 by the Armour Research Foundation for the Chicago Area Transportation Study. The original program assumed demands known, and assigned all traffic to the shortest paths between origins and destinations. The original program was revised and expanded as we shall describe below.

While this program was being developed in Chicago, similar efforts were underway in Washington at the Bureau of Public Roads in conjunction with the Washington Regional Highway Planning Committee and the General Electric Computer Department. The operation of the Bureau of Public Roads Traffic Assignment Program is described as follows:

"The selection of the quickest route from each zone to all others is the key to the assignment procedure ....

"With the zone to zone trips previously prepared and the minimum path route returned to its memory point, the computer loads the trips on the individual route sections comprising the minimum path routes between the zones. After the trips have been assigned from one zone, the computer selects the next zone and repeats the process. This is repeated until all of the trips from all of the zones have been loaded on route sections.

"At this point it is possible to examine the loads on the sections ....

"Some of the loads on individual links may exceed the capacity of the transportation facilities, thus affecting travel time or other criteria that were used to determine
the minimum time paths. In this situation ... new minimum paths are computed using a set of adjusted travel times. The automatic method for making these adjustments to the original network is called capacity restraint.

"After the trips from all zones have been computed and added to the network, the computer calculates the ratio of the assigned volume to the capacity for each route section and adjusts the travel time according to a predetermined relationship. This relationship says that the more the capacity of the link is used, the greater the travel time becomes." (19)

Zettel and Carll (20) summarize the twelve major metropolitan transportation planning studies underway in the United States by 1962. They summarize the revised Chicago traffic assignment program, which differs slightly from the Bureau of Public Roads program, as follows:

"Minimum path begins with free speeds on all routes. A capacity restraint is used after each zonal assignment. There is a specific ordering of zones to prevent distortion, but how the ordering is determined was not explained. Total flow on a link appears to be the same regardless of the order of zones, but interzonal travel times can be considerably different.

"The capacity function is based on a ratio of demand to average maximum capacity." (21)

Both the revised Chicago program and the Bureau of Public Roads program are based on the Charnes-Cooper game theoretical model of traffic equilibrium, which in turn is a special case of the Beckman equilibrium problem. Yet Charnes does not refer to Beckman and neither the Bureau of
Public Roads nor the Chicago Area Transportation Study report (22) refers to Charnes. It appears useful, therefore, to have demonstrated the similarity of these various formulations.

We remark a slight difference between the Bureau program and the Chicago program as described above. The Bureau program applies "capacity restraint", that is, calculates travel time as a function of loads, after all zones have been assigned. The Chicago program calculates travel time after each zonal assignment, thus more frequently distributing information regarding the state of the network. In our simple example in a later chapter, it will be seen that the latter method tends to reduce the instability of the solution by "forewarning players" of potentially bad decisions.

Finally, we return to our claim that this model is basic to the traffic assignment models used by metropolitan transportation planning organizations. For this point we cite the conclusion of Zettell and Carll (23) who point out that the assignment programs used at Chicago, Washington, Detroit, Pittsburgh, Toronto, Los Angeles, and Penn Jersey all rely on variations of the iterated minimum path assignment. These variations include: capacity restraint or none (the latter is equivalent to assuming constant travel times); "diversion curves" (the assignment of fractions of traffic to specified classes of roads, hence "diversion" from "normal" paths; the use of time and/or distance or cost as a measure of path length; and the diversion of flow to non-automobile modes of travel (in the Toronto program). These variations do not change the fundamental concept of the assignment program, which is the search for a stationary equilibrium flow pattern, based on the assumption that individ-
uals will seek minimum paths, coupled with an iterative technique for the evaluation of the effects of the interactions of individual strategies.

1.5 System Effects of Network Modification

1.5.1 The traffic assignment technique as an evaluative device

We have pointed out that in the Charnes model demand was held constant. In general, the constant demand traffic flow pattern is used in metropolitan transportation studies for determining the adequacy of the system for a given set of demands. Changes are made to the network and the new network is re-evaluated for its adequacy. In this way, a large number of design choices are evaluated for their effects on the total flow pattern, the so-called "system effects". New demands for transportation are calculated for a later time period. These demands are usually found by analyzing land use patterns, industrial and population growth rates, consumption functions, social preference patterns, etc. In the Penn-Jersey study, for example, the effects of changes to the transportation network are taken into account in calculating changes in land use and new demands for transportation. If locational preference is assumed to depend on travel time, following Alonso, these effects must be taken into account. On the other hand, in the Chicago Area Transportation Study, the effects of changes to the transportation network are not considered in determining land use patterns for the next period. This contrast is the subject of a current debate, further into which we need not go.
The essential point is that the minimum path assignment technique is used to simulate the flow of traffic for a given demand (although in the Toronto program, demands vary with path length; this results in a fairly involved iterative technique) in a manner which is considered to represent the actual behavior of vehicles on a network. The total response of the system to any changes in the design of the network may then be evaluated.

To quote the Traffic Assignment Manual:

"The purpose of traffic assignment may be listed as follows:

1) To determine the deficiencies in the existing transportation system by assigning the future trips to the existing system.

2) To assist in the development of a future transportation system through an evaluation of the effects of improvements and additions to the existing system.

3) To develop construction priorities by assigning the trips forecasted for intermediate years to their corresponding systems.

4) To provide systematic and reproducible tests for alternate system proposals.

5) To provide the highway designer with the design-hour traffic volumes." (24)

Recognition of the assignment problem as a means of evaluating total systems effects represents a significant advance over traditional traffic engineering approaches, which were generally restricted to considering the purely
local effects of any system improvement. An example of this traditional approach is that of Gardner (25), in which the still more traditional approach of scheduling highway improvements mainly on the basis of age and condition is modified to include priorities for congested highways. Such a local approach neglects the effect on total flow pattern which may result from simultaneous improvements to several roads. Such information might change the priorities which were obtained by considering the variation of only one parameter. An approach similar to that of Gardner is applied to urban streets by Haley, et al (26).

1.5.2 Combinatorial problems and mathematical programming

While the minimum path assignment program serves as a device for evaluating the total effects of changes to a traffic network, one quickly realizes the combinatorial problem involved in evaluating many proposed changes to a traffic network. Consider, for example, the problem of determining the best pattern of one way streets given a set of demands. If each street could be set either two-way or one-way in either direction, and there are no such streets, one would have \(3^n\) network to evaluate. For \(n = 12\), there are 531,441 networks to be evaluated. Clearly it would be too costly to evaluate all or even a significant fraction of the possible networks.

In order to provide a method for solving the problem of optimal improvements or changes to a network while avoiding the combinatorial problems involved, we return our attention to the other aspect of the traffic flow analysis of Beckman and Charnes and Cooper: the problem of system-optimal flow patterns. These authors observed the appropriateness of linear programming techniques for the solution of this prob-
lem. The main content of this thesis is to show that this method may be extended through so-called "integer programming" so as to deal in a computational convenient and conceptually natural way with the problem of finding optimal improvements to a traffic network.

The data for this technique are the same data as are applied to the traffic assignment programs: specification of demanded flow between pairs of points and specification of the capacity (or travel cost) function for each arc. It will be shown that a linear program can be formulated which gives an extremely good approximation to empirical data regarding average travel time per unit distance on a road of given characteristics, as a function of the total flow on that road per unit time.

In any improved network, the flow solution will not be determined by the minimum path principle, but by the principle of minimum total travel time for all vehicles operating on the system.

If one objects that this is not a "realistic solution", that vehicles would not tend to behave so as to minimize total travel time, we would reply that this method provides a bound on the quality of solution which can be obtained. That is, no minimum path "predictive" solution will ever correspond to a more efficient utilization of facilities than the solution provided by the linear programming techniques. This technique therefore provides a point of reference for the evaluation of any "predictive" solution.

From another point of view the linear program is seen as a "normative" solution; it demonstrates the pattern to be desired all other costs being the same. In this respect
it will be shown to yield valuable information on policies for control of traffic leading to efficient utilization of the existing facilities. The decomposition algorithm of linear programming will be interpreted so as to yield this control information. This algorithm also provides a computational technique which permits the solution of problems pertaining to extremely large networks.

Finally, the difference between the "normative" and the "predictive" solutions can be reduced (or even eliminated) under certain conditions. Jorgenson (27) showed that where travel costs are constant, (that is, congestion effects are absent), and no route is loaded to absolute capacity, the minimum path solution to a given problem will be identical to the minimum total system cost solution. This suggests that in designing optimal rural traffic systems, where congestion effects are likely to be absent, the linear programming solution may be considered to a "predictive" solution.

On the other hand, in urban planning, where congestion effects are all too significant, any given flow pattern may be approximated by constraints bounding the flow in the linear programming model. This means that an optimum solution will be found, subject to the constraint that the flows will not be very different from the flows specified as data. This is a means of using linear programming so as not to conflict with various assumptions about "unavoidable", or known, flows in certain parts of the network.

1.6 **Summary**

Beginning with classical land economics, we rapidly traced the evolution of the idea of transportation cost as a
determinant of location. The modern theories of Beckman, Wingo and Alonso have been shown to underlie the traffic planning models used in current American metropolitan transportation planning. Two forms of the model were recognized: the minimum path aspect, and the total system optimum aspect. The former is calculated by an iterative technique which allows one to evaluate the total effects of changes to the transportation system, but becomes extremely costly when many variations are to be evaluated. The latter aspect is calculated by a linear programming technique; we shall demonstrate that the technique can be extended to determine optimal improvements to a transportation system, at a nominal cost of calculation.

In the following chapter we shall discuss in complete detail the formulation of the basic analytical model employed in this study. Following Charnes and Cooper, we shall use a "multi-copy flow model". This formulation has appeared in a number of contexts and in Appendix A we shall give a comprehensive review of the applications of this and related models. Subsequently, we shall present and illustrate our models for optimal improvements to traffic networks. And finally we shall study the implications of the decomposition algorithm of linear programming for network control.
1.7 Notes to Chapter I


(2) ibid., p. 4.

(3) ibid., pp. 6-7.

(4) ibid., p. 9.


(7) M. Beckman et al, op. cit., p. 61.

(8) ibid., pp. 80-81.

(9) ibid., pp. 86-87.


(13) ibid., p. 184.


(18) op. cit., p. 1-3.

(19) ibid., p. II-3.


(21) ibid, p. 58.

(22) Chicago Area Transportation Study, *Final Report*.


CHAPTER II
Development of a Mixed Integer Mathematical Programming Formulation for Determining Optimal Improvements to a Traffic Network.

2.1 Introduction

In this chapter we shall develop in detail a mathematical model for determining both the optimum flow pattern on a vehicular network with known demands, and the optimal changes to be made to the network.

We shall develop, in parallel with a general formulation, a particular model of a section of a metropolitan arterial road system. The computational examples to be discussed in the following chapter will be based on this particular model. We would like to emphasize at the outset that no loss of generality of the method is incurred by directing our discussion primarily to the particular model. We employ this model to illustrate an application of the general concepts.

The model developed in this chapter is basically an extension of the Charnes-Cooper multi-copy traffic flow model (1).

Our model differs from the Charnes-Cooper model in the following respects:

1) We assume travel time on a road in each direction to depend on the total flow on that direction, independent of flow in the opposite direction. This ignores the effect of the interference caused by cars overtaking and passing, on flow in the opposite lane. This for-
mulation is therefore suitable for modelling urban streets and expressways, rather than, for example, two lane rural highways. In particular this formulation will be shown to be appropriate to a model for it determining the optimum configuration of one-way streets. Charnes-Cooper assume travel time on a link in either direction to be a function of total flow in both directions.

2) Charnes-Cooper associate a network "copy", or set of flow conservation equations, with each single origin and all its associated destinations. We generalize this to associate a copy with any collection of origins and destinations with either at most one origin or at most one destination in the set. In a linear programming formulation there is only a formal difference between our formulation and that of Charnes-Cooper. However, we lose the correspondence between the Charnes-Cooper game theoretical model, in which a player is associated with each origin, and the linear programming model in which a "network copy" is associated with each origin. In other words, our copies can be associated with origins, or with destinations.

3) In order to consider the problem of network improvement, we transform the capacity functions into variables. In the Charnes-Cooper model, capacity functions are fixed.
In addition, we analyze the nature of the capacity function in great detail. Charnes-Cooper simply assume the capacity function to be a convex function which can be approximated using a linear program.

2.2 Physical Statement of the Basic Problem

2.2.1 The general physical problem

We are given a network of roads, consisting of nodes (interchange points) and road links connecting the nodes. We define a link as a road channel permitting flow in one direction only. A two-way street corresponds to a pair of links. We are told how many vehicles enter at each point of the network and we are told their respective destinations. The required flow between an origin node and a destination node, in terms of vehicle per hour is known as the transfer volume. The plotting of transfer volumes on a map of origins and destinations is called the desire line pattern.

Experimental evidence leads us to infer that the average velocity on a road link decreases as the total number of vehicles per hour, \( f \), (flow rate) travelling on that link increases. In other words the average vehicular travel time for a road link is an increasing function of the total vehicular load per unit time on the link. We call this function the average travel time function, \( t(f) \) (8). The form of the function depends on the number of signalized intersections on the link, the number of lanes, the nominal speed limit, the road geometry, and the composition of traffic on the link (2).
We assume that the average travel time function for each link is given. This function relates the travel time in minutes per mile, per lane, per vehicle, to link loading in vehicles per hour. We assume that each link is completely characterized by its average travel time function, so that we can put aside all the detailed data from which this function was derived.

The total time spent traversing a link by all vehicles is the total time cost, $T(f)$, for the link; it equals the volume on the link times the average travel time for that volume on the link and is a function of link loading ($T(f)=f \cdot t(f)$). We define "system time cost", $T$, as the sum over all links of the link total time costs. It is the total time spent on the network by all vehicles traversing the network. See section 2.4.2 for dimensions.

Given the average travel time function for every link, and the desire line pattern, we are asked to minimize the system time cost. We permit ourselves two basic control techniques for obtaining the objective. First we are to specify the routes which are to be used in transporting the transfer volumes through the network, and the number of vehicles using each route. Secondly, we may designate certain streets as one-way streets if this will help reduce the system time cost. Later we shall include the option of constructing additional links, or widening existing links.

2.2.2 The particular form of the physical problem

We shall consider a sample network which is to be construed as representing an arterial street network which lies between a central business district and the entrance
to a freeway. (See Figure 2.2.2 A) We shall suppose the required flow of traffic will represent rush hour flow from the CBD to the freeway entrance (radial flow) and in the two directions perpendicular (circumferential flows). (See Figures 2.2.2 B and C for two characterizations of rush hour flows. The second is more involved and we shall use it as a basis for our examples.) The network has 9 nodes. Nodes 1, 2, 3 lie at the edge of the CBD, node 8 is the freeway entrance. (See Figure 2.2.2 D) Generally, a pair of oppositely directed links connects each pair of nodes.

Instead of specifying the length, number of intersections, number of lanes, and all the other factors which determine the mean travel time function for each link, we shall subsume all these factors in the specification of the travel time function itself for each link. This average travel time function has been determined experimentally for various road types (3). We shall suppose all traffic to be automobiles exclusively, all links are to consist initially of one lane in each direction, and to be arterial roads one mile long, with a speed limit of thirty miles per hour. Each link is assumed to have 3 signalized intersections per mile. An average travel time function characterizes such a link, and we shall assume all links to be initially identical. These choices do not restrict the generality of the solution; they are designed simply to make a convenient illustrative example.

In order to represent the flow pattern of Figure 2.2.2 C in a network model, we proceed as follows.
FIGURE 2.2.2A  CENTRAL BUSINESS DISTRICT AND FREEWAY ENTRANCE
FIGURE 2.2.2B  CHARACTERIZATION A OF RUSH HOUR FLOW PATTERN
FIGURE 2.2.2C  CHARACTERIZATION B OF RUSH HOUR FLOW PATTERN
LEGEND:

1 NODE, (INTERSECTION)

ROAD LINK

FREEWAY

NETWORK OF ARTERIAL ROADS

FIGURE 2.2.2D SHOWING NODE AND LINK NUMBERS
FIGURE 2.2.2E DESIRE LINE PATTERN CORRESPONDING TO CHARACTERIZATION B
LEGEND:

- COMMODITY X TRAFFIC INPUT NODE
- COMMODITY X TRAFFIC OUTPUT NODE

PATTERN OF INPUTS AND OUTPUTS

FIGURE 2.2.2F CORRESPONDING TO CHARACTERIZATION B
The transfer volume set consists of three components. The first is flow demand from nodes 1 and 3 to node 8. This flow can be considered the mainstream radial flow from the CBD to the freeway entrance. There are two flows transverse to the radial flow. One transverse flow runs from node 7 to node 3, the other runs from node 9 to node 1. We have defined the transfer volume as the number of vehicles leaving a particular origin node for a particular destination node. We have characterized three flows: a mainstream flow and two opposing transverse flows. (See Figures 2.2.2 B and C.) The mainstream flow is made up of vehicles from two origins heading for the same destination. This flow must be distinguished from the crossstream flows and the crossstream flows from each other. Following the terminology of Ford and Fulkerson (3) we shall refer to the various flows as commodities. It is clear that a commodity may be made up of several transfer volumes; a commodity is not necessarily the flow between one origin and one destination node. The mainstream flow, from nodes 1 and 3 to node 9, forms the first commodity (A). The crossstream flows, from node 7 to node 3 and from node 9 to node 1, form the second and third commodities (B and C) respectively. (See Figures 2.2.2 F and G.)

A commodity may also be interpreted as a flow of a particular vehicle type. Two different commodities with the same origin-destination points might represent, for example, the flow of cars and the flow of trucks between these points. It is possible (and advisable) to specify different travel time functions for different vehicle types. While this interpretation of commodities may be directly implemented with the model we are constructing, we shall restrict ourselves to problems involving only automobile flows.
By loading the mainstream and the two crossstream flows on the network we have flows which are essentially at right angles, and others which are in opposite directions. These properties are intended to illustrate the characteristic effects of one way street patterns on general flow requirements.

### 2.3 Conditions Required to Specify the Physical Problem

From the verbal description of the physical problem, we shall construct a mathematical apparatus which will represent the problem conditions and enable us to reach the goal of the problem by analytical means. The mathematical representation must contain the following categories of information:

(a) **system operating conditions:**
   1. average link travel time as a function of loading
   2. flow limits for links (link capacities)
   3. network configuration (specification of the connections of nodes and links)

(b) **problem requirements:**
   4. the commodity input-output data at each node, derived by grouping transfer volumes

(c) **the problem objective:**
   5. determination of the minimum system time cost and optimal improvements to the network

The following sections will show how the mathematical representation of this information is constructed.
2.4 Transition from Empirical Data to a Theoretical Representation: Analysis of the Link Average Travel Time Function and the Link Capacity Limit

2.4.1 Introduction

The conditions and requirements enumerated in section 2.3 define the problem. They state the conditions to be satisfied (the commodity flow pattern), the operational properties of the system (the link travel time functions, the link capacity limits, and the network connections) and a goal for the analysis. We shall show how the conditions to be satisfied and the operational properties of the system can be represented by a set of linear inequalities and equations. The goal for the analysis is the minimization of a system time cost function, and an optimal network improvement scheme, for example, the determination of the optimal one-way street pattern (which will be called the switching pattern).

The mathematical technique of minimizing a function while satisfying a set of inequalities and equations which limit the permissible range of the arguments of that function is called mathematical programming. If the limiting, or constraining inequalities and equations, are linear and the function to be minimized (called the objective function) is also linear, the technique is called linear programming (4). The problem itself is called a linear program.

There exist efficient codes for electronic computers for the solution of linear programs; therefore it would be greatly to our advantage if the problem turned out to be soluble by this technique. The computer codes employ a computing algorithm called the "simplex method", or its variants.
The burden of this section will be to show that our problem can be set up in such a way that the objective function becomes linear. There will be no difficulty in showing later that the constraining inequalities and equations are linear, so that constructing a linear objective function is the key to obtaining a linear programming formulation for the problem.

First we show that the objective function in its direct form is not a linear function.

The objective function to be minimized has the following form:

System time cost = \( T = \text{sum over all links (total travel time function for each link)} \)

The total travel time on a link is a function of the total flow on a link. We must distinguish among flows belonging to different commodities so that the flow on a link is the sum over all commodity flows on the link. Thus the objective function has the form:

\( T = \text{sum over all links (function of total flow on each link)} \).

We shall next analyze the properties of the link total travel time functions. We shall show that their properties generate a non-linear system time cost, but that a linear approximation can be constructed.

2.4.2 The empirical link average travel time function

Irwin and von Cube (5), of the Traffic Research Corporation, have derived a family of travel time functions for arterial streets in Toronto. Each curve is character-
ized by the number of signalized intersections per mile, the nominal speed limit, the composition of traffic on the link (i.e., automobiles, and/or streetcars and/or trucks), and whether the link is an artery or a freeway.

Figure 2.4.2 A (after (5) ) shows empirical data for an arterial road where designed speed limit is 30 mph. Average travel time in minutes per mile per vehicle is plotted against traffic volume (or flow rate) in vehicles per lane per hour. The empirical data have been approximated by a piecewise-linear function, which we shall call the empirical average travel time function, \( t(f) \), or simply the empirical average. We call this approximation "interpretation 1". To simplify our discussion without sacrificing generality we can consider links of one mile length, one lane only, loaded for one hour, so that \( t(f) \) would be measured in minutes/vehicle and \( f \) in vehicles.

We see that on the abcissa are marked the points \( f_c \) and \( f_m \), or the critical flow and the "maximal flow". On the ordinate are marked three special values of \( t(f) \): \( t_o \), or "zero-flow travel time", \( t_c \), or "critical flow travel time", and \( t_m \), "maximal flow travel time". From Figure 2.4.2 A, \( t_o = 3.0 \) min/mi, corresponding to 20 miles per hour; \( t_c = 3.5 \) min/mi, corresponding to 17.2 miles per hour, and \( t_m = 6 \) min/mi, corresponding to 10 miles per hour.

The travel time \( t_o \) represents the average time for a single vehicle to traverse the link, in the absence of any other vehicles. We see that as the load increases to \( f_c \), 500 vehicles per hour, the average travel time increases very slowly, at rate \( d_1 \). For this load range, the piecewise linear approximation of Figure 2.4.2 A fits the observed points very well. When the load passes the critical
FIGURE 2.4.2A

CAPACITY FUNCTION, FOR ROADS WITH CARS ONLY

30 MPH, 3 SIGNALS/MILE, (TRC), INTERPRETATION 1
flow, \( f_c \), 667 vehicles per hour, the average travel time increases rapidly, at a rate \( d_2 \). Beyond the "maximum" flow, \( f_m \), the average travel time increases still more rapidly, at rate \( d_3 \). In this range, the observed points are not close to the linear approximation.

In the following discussion, we shall derive a piecewise linear approximation to the empirical average travel time function, which differs from that used by Irwin and von Cube. This approximation will involve a redefinition of the "maximal" flow. We shall obtain this by relating the average travel time data to corresponding vehicular densities. We shall be led to the following conclusions:

(a) \( f_m \), the "maximal flow on the link" according to Irwin, should not in fact be considered a limit on permissible flow values,

(b) the Irwin and von Cube piecewise linear function (interpretation 1) extends beyond its permissible domain of definition. In other words, the effective maximal flow value, which we shall call \( f_n \), lies above \( f_m \) but below the largest flow, \( f = 800 \), for which the Irwin function is defined.

2.4.3 Reinterpretation of the empirical link average travel time function

To analyze the notion of "maximal flow", \( f_m \), we shall compare our analysis which is based on the data for \( t(f) \) mean travel time as a function of volume, to a related notion, "the fundamental diagram of road traffic" (6).

First let us look somewhat more closely at the observed data on which \( t(f) \) is based, in Figure 2.4.2 A. We note the
following characteristics of Irwin's piecewise linear approximation to the experimental data.

(a) For flow rates up to 400 vehicles (per hour, per lane - these dimensions to be implicit in the following discussion) the experimental points are quite close to the piecewise linear function as drawn by the Traffic Research Corporation.

(b) As the flows approach $f_c$, the goodness-of-fit begins to deteriorate, and beyond $f_c$, the fit is noticeably worse.

(c) At $f_m = 667$ vehicles and beyond, the three remaining observed points are widely skewed from the piecewise linear function $t(f)$.

We propose two contrasting interpretations for this.

First, we may consider $f_m$ as the flow volume at which congestion, or turbulence in the stream of vehicles sets in. Then flows greater than $f_m$ represent conditions of disturbed flow, hence the corresponding mean travel times are erratic.

Secondly, we may refer to Figure 2.4.3 A, which is also taken from the Traffic Research Corporation study (5). In this theoretical diagram of average travel time versus flow rate, the actual flow is shown approaching a maximum possible value, $f_m$. Were demanded flow to increase beyond $f_m$, the actual flow would decrease due to congestion and the average travel time would continue to increase. That is the meaning of the reversal and return of the average travel time curve in this Figure.

This diagram suggests interpretation 2 of Figure 2.4.2 A as shown in Figure 2.4.3 B. By flattening the curve we pass to interpretation 3, shown in Figure 2.4.3 C.
Figure 2.4.3A Volume-Time Relationships for Typical Road Sections
FIGURE 2.4.3B  CAPACITY FUNCTION, INTERPRETATION 2
FIGURE 2.4.3C  CAPACITY FUNCTION, INTERPRETATION 3
Suppose the correct approximation to the mean travel time data were not the three piece linear approximation as drawn by Irwin (the broken lines in Figures 2.4.3 B and C), but instead should be the represented by the solid curve of Figure 2.4.3 C.

Note that the points corresponding to flows of 740 and 760 vehicles (point 1 and point 2) lie very close to the second piece of the (almost) linear approximation. Note also that point 3, corresponding to a flow of 667 vehicles, seems to represent a flow beyond the maximal flow, $f_m$, as suggested by the theoretical diagram in Figure 2.4.2 B. In other words, point 3 appears to lie on the returning part of the theoretical average travel time curve. Interpreted in this manner, the data would suggest that the maximal flow lies somewhere at or beyond a flow of 760 vehicles.

As there are only three data points in the region of flow - 760 vehicles and beyond, it is difficult to decide on the basis of limited evidence which interpretation is correct.

2.4.4 Flow density and flow rate

In an attempt to obtain more information on which to base our decision, we shall relate our average travel time function to the "fundamental diagram".

First we summarize the theoretical derivation of the fundamental diagram, following Haight (6).

The basic kinematic equation for a vehicle is

$$s = mt,$$
where \( s = \) distance travelled per vehicle
\( m = \) mean speed
\( t = \) time travelled per vehicle.

This same equation can be put in the form:

\[
\frac{1}{t} = m(\frac{1}{s}).
\]

Since \( t \) had the dimension time per vehicle, its reciprocal has the dimension vehicles per time, or flow rate.

Hence we set

\[
f = \frac{1}{t} = \text{flow per unit time} = \text{volume}.
\]

Also, \( s \) had the dimensions of distance per vehicle. Its reciprocal has the dimensions vehicles per distance. Therefore \( \frac{1}{s} \) is the (mean) density of vehicles on the road.

We set

\[
d = \frac{1}{s} = \text{density of vehicles per unit length of road}.
\]

Substituting \( f \) and \( d \) into the second form of the basic kinematic equation we obtain

\[
f = md.
\]

That is, flow per unit time equals mean speed times density of vehicles per unit length of road.

We have empirical evidence that the increased density of vehicles on a road leads to decreased average speeds, so that we write mean speed and consequently flow volume as functions of density:

\[
f(d) = m(d) \cdot d
\]

Plotting the flow rate against density gives the "fundamental diagram".
To estimate the shape of this curve we use the following points:

(a) There is an upper limit on \( d \). That is, the maximum density of vehicles on a road is clearly limited by the minimum vehicle length. For example, if the minimum car length is fifteen feet, we cannot have \( d \) greater than \( d^1 = 1/(15) \), or one car per fifteen feet.

(b) Jam packed traffic is essentially stationary, so that the flow at maximum density is zero. Also, if there are no cars (zero density), the flow must be zero. Hence
\[
f(0) = f(d^1) = 0.
\]

(c) For low densities, the traffic moves at a "mean free speed", the speed at which one vehicle would move if it were alone on the road. As density increases, interference among vehicles (or "turbulence") disturbs the free flow and mean speed begins to decrease. Consequently the flow rate begins to decrease. At a certain density the flow rate reaches its maximum, which we call the "capacity" of the road. Further increases in density being the flow rate down to its limit of zero.

These considerations lead to the fundamental diagram shown in Figure 2.4.4 A.

Note that the slope of \( f(d) \) has the dimensions of speed
\[
f/d: (\text{cars/time})/(\text{cars/distance}) = (\text{distance/time}).
\]

In fact the derivative of \( f(d) \) is
FIGURE 2.4.4A  "FUNDAMENTAL DIAGRAM OF TRAFFIC FLOW"
\[ \frac{d f(d)}{d(d)} = d \cdot \frac{d m(d)}{d(d)} + m(d) \]

At \(d = 0\), the derivative has the value \(m(0)\), so that the initial slope of \(f(d)\) is in fact the mean free speed.

2.4.5 Relating density, flow rate, and average travel time

Having defined the notion of flow as a function of density, \(f(d)\), we proceed to relate it to the data for average travel time as a function of flow. In the following we show how one can determine \(f(d)\) from the \(t(f)\) data.

An example will illustrate the point. Suppose we have a \(t(f)\) function according to which the average travel time \(t\) for a flow rate of 540 vehicles per hour is given as 6 minutes per mile. (This corresponds to an average speed of 10 miles per hour). We can calculate the mean density of vehicles as follows.

Consider a queue of vehicles lined up to pass over a one mile strip. The vehicles in the queue are moving at an average speed of 10 miles per hour. How many miles long is the queue so that the last vehicle reaches the end of the required strip just one hour after the first? (See Figure 2.4.5 A.) The last vehicle is travelling at ten miles per hour, it therefore must be ten miles from the end of the required strip. The first vehicle is one mile from the end of the required strip.

Thus the queue is \(10 - 1 = 9\) miles long. Since we assumed a flow rate of 540 vehicles passing over the required strip each hour, corresponding to a speed of 10 miles per hour, there are 540 vehicles in the nine mile queue.
FIGURE 2.4.5A  CONSTRUCTION TO ILLUSTRATE DERIVATION OF $f(d)$ FROM $t(f)$
The density is therefore 60 vehicles per mile, or one vehicle every 88 feet.

Now we can algebraically relate the density to the flow rate by means of the travel time function.

The queue length is

\[ L(f) = \left( \text{miles per hour corresponding to } f \right) \cdot \left( \text{1 hour} \right) - 1 \text{ mile.} \]

The quantity "miles per hour corresponding to the flow rate } f" is simply the mean speed \( V \) as a function of } f. But \( v(f) \) in miles per minute is simply

\[ v(f) = \frac{1}{t(f)} \text{ in miles per minute} \]

and

\[ v(f) = \frac{60}{t(f)} \text{ in miles per hour.} \]

Therefore the queue length is

\[ L(f) = \left( V(f) \cdot 1 \right) - 1 = \left( \frac{60}{t(f)} \right) - 1 \text{ in miles} \]

and the density of vehicles at flow rate } f is

\[ d(f) = \frac{f}{L(f)} = \frac{f}{\left( \frac{60}{t(f)} - 1 \right)} \]

\[ \begin{align*}
  d(f) & \text{ in veh/mi} \\
  f & \text{ in veh/hr} \\
  t(f) & \text{ min/mi/veh.}
\end{align*} \]

This is the relation between density as a function of flow and travel time as a function of flow which we have been seeking. The relation \( d(f) \) is shown in Figure 2.4.5 B, a reflection of the fundamental diagram of Figure 2.4.4 A about the 45° line.

Since the function \( t(f) \) is two valued, (as in Figure 2.4.3 A) one asks the question: for which part of the
FIGURE 2.4.5B  REFLECTED "FUNDAMENTAL DIAGRAM"
diagrammed curve is the above expression for \( d(f) \) valid. In the following discussion we shall answer this question, while reconciling the values obtained for \( d(f) \) using \( t(f) \) with the "theoretical form" of \( d(f) \) obtained from Figure 2.4.4 A. The following paragraphs analyze the functions \( d(f) \) obtained from the two different interpretations of the mean travel time function \( t(f) \).

If we use interpretation 1 of the capacity function \( t(f) \) (Figure 2.4.2 A), the three piece linear fit to the data, we obtain \( d(f) \) as plotted in Figure 2.4.5 C. In this figure, the density begins at zero, and grows at any increasing rate to the end of the range of definition for \( t(f) \), \( f = 800 \). We have extrapolated the curve and we see that \( d(f) \) hits its ceiling (based on a density of 1 car per 20 feet) at \( f = 825 \), \( d = 264 \).

We have several contradictions to resolve when comparing this curve to the diagram of Figure 2.4.5 B. The \( d(f) \) curve obviously does not fold back on itself in Figure 2.4.5 B. At maximum density we expect the flow to be zero, but according to this curve the corresponding flow rate is 825. Clearly the expression \( d(f) \) is not valid for flows greater than some unknown limiting flow \( f_n \). We might set \( f_n = f_m \), thus defining \( f_m \), the flow at which turbulence sets in, as the maximal flow. Then we would construct a reflected "fundamental diagram" plotted as the solid line in Figure 2.4.5 D. This construction is developed by plotting the density function of Figure 2.4.5 C, cutting off the curve at \( f = f_m \), and reflecting the curve through the cutoff point about the horizontal axis. This would have roughly the form postulated for the transposed fundamental diagram (Figure 2.4.5 B). There are two dis-
Figure 2.4.5C  DENSITY $d(f)$ based on $t(f)$, Interpretation I
FIGURES 2.4.5D & 2.4.5E 
REFLECTED FUNDAMENTAL DIAGRAMS 
FOR LIMITING FLOW RATES $f_m$ AND $f_n$
crepancies, however. The slope of \( d(f) \) at \( f_m \) is far from infinity, and the density \( d_a \) corresponding to \( f_m \) is far below the midpoint \( d_o \) of the range of possible densities. Furthermore point 3 appears to have no relation to the curve.

These points suggest that we have assigned the maximal flow at a value which is too low: the function \( d(f) \) should be allowed to increase more, before being forced back on itself. (See Figure 2.4.5 E.) In this case we assign of maximum flow \( f_n \) greater than \( f_m \), and obtain a correspondingly greater density.

Suppose we re-interpret the capacity function \( t(f) \) by extending the second linear piece through values of \( f \) equal to 760, as shown by the solid line in Figure 2.4.5 F. This line approximates the points 1 and 2. We terminate the definition of \( t(f) \) at \( f = 760 \). This is equivalent to defining the effective maximal flow as \( f_n = 760 \).

Somewhere between \( f_n = 760 \) and \( f_p = 825 \), by the density argument, \( f \) must take its actual maximal value. We assume that point 3 corresponds to a flow greater than the actual maximal value. The solid line in Figure 2.4.5 F gives interpretation 4 of the capacity function. The dotted line shows interpretation 1, where it differs from interpretation 4. Interpretation 4 can be considered as derived from interpretation 3 by further flattening the curve and bounding the flow domain.

We plot in Figure 2.4.5 G the density function \( d(f) \) resulting from this interpretation of the travel time data. In particular \( d_a = d(667) \) was checked by direct calculation using the queue length argument.
FIGURE 2.4.5F  CAPACITY FUNCTION, INTERPRETATION 4
The resultant plot (Figure 2.4.5 G) has certain features which seem to be an improvement over the \( d(f) \) plot (Figure 2.4.5 C) derived from the first interpretation.

The most significant property is that points 2 and 3 appear to lie on the return part of the \( d(f) \) curve. Point 3 is obviously above and to the left of the return point.

The reflected fundamental diagram, or "theoretical density curve" is also plotted in Figure 2.4.5 G. It is a parabola which meets the density axis at \( d = 0 \) and \( d = d_1 = 264 \); the slope is vertical at a maximum flow value \( f_p = 825 \), corresponding to a midpoint density \( d_\circ = 132 \).

The \( d(f) \) curve derived from interpretation 4 of the capacity function \( t(f) \) suggests a "parabola" whose return point occurs at a maximum flow \( f_n = 760 \) with a "midpoint" density of 108.

Adopting interpretation 4 therefore gives a reasonable approximation to the theoretical density curve. The return point occurs for a density which is not far from the theoretical midpoint, and a maximum flow not far from the theoretical maximum.

It generates a density curve which is much closer to the theoretical curve than the density generated by interpretation 1, either as given by Irwin (Figure 2.4.5 C) or truncated at \( f = f_m \) and reflected (Figure 2.4.5 D).

These considerations suggest that the maximum flow lies between \( f_n = 760 \) veh/hr (approximation to data) and \( f_p = 825 \) veh/hr (theoretical maximum density), and not at \( f_m = 667 \) veh/hr (maximal flow according to interpretation 1).
FIGURE 2.4.5G

- DENSITY BASED ON CAPACITY FUNCTION INTERPRETATION 4
- DENSITY BASED ON CAPACITY FUNCTION INTERPRETATION 1
- REFLECTED FUNDAMENTAL DIAGRAM, THEORETICAL
In our computational model, we shall assign a "practical capacity" \( f_n \) limiting total flows on links to the range

\[ 0 \leq f \leq (f_n = 760 \text{ vehicles/hour}) \]

The empirical average travel time function \( t(f) \) in minutes (or simply the empirical average) is now given by:

\[
t(f) = \begin{cases} 
  t_o + d_1 f & 0 \leq f \leq f_c = 500 \text{ vehicles} \\
  t_c + d_2 f & f_c \leq f \leq f_n = 760 \text{ vehicles} 
\end{cases}
\]

\[ t_o = 3.0 \quad t_c = 3.5 \]

\[ d_1 = 0.0010 \quad d_2 = 0.0150 \]

This is to represent average travel time per vehicle on an arterial link of length one mile, with three signalized intersections, one lane, loaded for one hour.

The limitation of flows to values not exceeding 760 vehicles will be referred to as "the link capacity constraint".

2.5 From Average to Total and Incremental Travel Time Functions

In the previous section we analyzed average travel time data in terms of piecewise linear approximations. We determined the implications of various interpretations of the data for the related "fundamental diagram" of traffic flow. This led us to revise our original interpretation of the data and to accept a two-piece linear approximation to the average travel time.
We stated that the goal of our mathematical programming technique is to minimize total travel time on the network, which is the sum of the total travel times for each link. The input to the linear program will be incremental (or marginal) travel time data - the unit increase in travel time to the system per vehicle added.

In the following sections we analyze the relations of these functions.

2.5.1 The empirical total travel time function for a link

In the previous section, we derived an empirical average travel time function $t(f)$ for a flow on a road link of length one mile, loaded for one hour. This is a two-piece linear function designed to approximate the empirical data of Figure 2.4.5 F.

From $t(f)$ as given in Figure 2.4.5 F, we can construct an empirical total travel time function $T(f)$ given by

$$T(f) = f \cdot t(f)$$

for steady state flow

where $f$ is given in vehicles, $t(f)$ in minutes/vehicle, and $T(f)$ is in minutes.

The function $T(f)$ gives the total travel time spent on the link as a function of one hour's loading at a fixed volume. The function $T(f)$ is shown in Figure 2.5.1 A as the solid curve.

Note that $T(f)$ is not a linear function, although it evidently can be approximated by a two-piece linear function, the dotted curve in Figure 2.5.1 A.

Since the objective function, or total travel time over the system, was defined as the sum over all links of
Figure 2.5.1A: The Empirical Total Travel Time Function for a Link

- Slope $\alpha_2 \approx 15.5$
- Slope $\alpha_1 \approx 3.2$
the link travel time functions, the objective function is not a linear function either, and we do not yet have a linear program.

2.5.2 The approximate total travel time function for a link and the arc-pair concept

We shall now show that it is possible to approximate the empirical link total travel time function \( T(f) \) and to transform variables in such a way that the objective function does become linear.

We observed in Figure 2.5.1 A that the empirical total travel time curve can be approximated quite closely by two linear segments. The first segment extends over the flow range 0 to \( f_c \), the second from \( f_c \) to \( f_n \). We shall call these ranges range 1 and range 2 respectively. The approximation was drawn so as to match the empirical function at flows 0 and \( f_n \), to be continuous at \( f_c \), and to minimize the deviation at intermediate points. We measure the slopes of the pieces in the approximation and call these slopes \( a_1 \) and \( a_2 \), with \( a_1 \) less than \( a_2 \). We give the name \( T(f) \) to this two piece linear approximation to the empirical link total travel time function \( T(f) \).

The value of \( T(f) \) is given by

\[
T(f) = \begin{cases} 
    a_1 f & \text{for } 0 \le f \le f_c \quad \text{(range 1)} \\
    a_1 f_c + a_2 (f-f_c) & \text{for } f_c < f \le f_n \quad \text{(range 2)} 
\end{cases}
\]

\( T(f) \) is the total travel time in minutes, \( f, f_c, f_n \) are flows in vehicles.
We next show that by introducing two new variables \( f_1 \) and \( f_2 \) whose sum equals \( f \), we can construct a linear function for \( T(f) \). The variables \( f_1 \) and \( f_2 \) may be visualized as flows parallel to, and in place of, the original flow \( f \). The flow \( f \) is split into the flows \( f_1 \) and \( f_2 \).

Suppose we restrict the range of \( f_1 \) between 0 and \( f_c \), and the range of \( f_2 \) between 0 and \((f_n - f_c)\).

We define two new cost functions

\[
T_1(f_1) = a_1 f_1
\]

\[
T_2(f_2) = a_2 f_2
\]

Suppose the original mathematical program, as it attempts to satisfy its conservation of flow equations and to minimize the objective function, decides that a flow of value \( f \) should be sent on the link in question. Suppose we tell this program that it may not use the flow \( f \) (which has cost function \( T(f) \)), but instead may use any combination of flows \( f_1 \) and \( f_2 \) which sum to \( f \), provided the limits and cost functions for \( f_1 \) and \( f_2 \) are observed.

The mathematical program, which wants to minimize cost, realizes that as much as possible of the flow \( f \) should be sent via \( f_1 \), since the cost coefficient \( s_1 \) is less than \( s_2 \). Thus, for \( f \) less than or equal to \( f_c \), all of \( f \) is sent via \( f_1 \) and the total cost is

\[
T(f) = T_1(f) = a_1 f \quad \text{for } 0 \leq f \leq f_c
\]

Suppose now that \( f \) lies between \( f_c \) and \( f_m \). First, as much as possible of \( f \) is sent via \( f_1 \); this is a flow of value \( f_c \) at a cost \( a_1 \). The remainder of the flow is \((f-f_c)\), and is sent via \( f_2 \) at a cost \( a_2 \) \((f-f_c)\) so that the total cost is
\[ \mathcal{T}(f) = a_1 f_c + a_2 (f - f_c) \quad \text{for } f_c \leq f \leq f_n \]

But referring to the original cost function \( \mathcal{T} \), which was arrived at by splitting the flow \( f \) between two new flows \( f_1 \) and \( f_2 \), is the same as \( \mathcal{T}(f) \).

We can summarize the construction as follows: Suppose we replace the original link by a pair of fictitious links which we shall call arcs. The flow on arc 1 is called \( f_1 \) and its value must not exceed \( f_c \), the critical flow for the original link. The flow on arc 2 is called \( f_2 \), its value must not exceed \( (f_n - f_c) \), the difference between the maximum and the critical flows on the original link. The cost coefficient for \( f_1 \) is \( a_1 \), which is the approximate slope for the original link's total cost function for flows less than \( f_c \). The cost coefficient for \( f_2 \) is \( a_2 \), which is the approximate slope for the original link's total cost function for flows between \( f_c \) and \( f_n \). The flows \( f_1 \) and \( f_2 \) sum to \( f \). Moreover, the attempt to minimize total travel time means that as much of flow \( f \) as possible will be allocated to \( f_1 \), the new flow variable with the smaller cost coefficient. That is, \( f_2 = 0 \) if \( f \) is less than \( f_c \). Thus

\[ \mathcal{A}(f) = a_1 (f_1) + a_2 (f_2) \]

is a linear functional, and will equal \( \mathcal{T}(f) \) when flow costs are minimized.

Suppose we rewrite the problem parameters enumerated in section 2.3 in terms of arc pairs. We replace every link in the network by a pair of arcs with flow limits and cost coefficients given by the rules of the preceding paragraph. These rules are summarized in Table 2.5.2 (1). We now have a problem which is equivalent to the original problem in the sense that we can solve the new problem, add the flows
Replace each link by a pair of arcs having the same directional orientation as the link. Define the arc capacities and travel time functions as follows:

<table>
<thead>
<tr>
<th></th>
<th>link</th>
<th>odd arc</th>
<th>even arc</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>critical flow</strong></td>
<td>$f_c$</td>
<td>$(f_c)$</td>
<td>$(f_n - f_c)$ (implicitly equal to capacities)</td>
</tr>
<tr>
<td><strong>maximal flow</strong></td>
<td>$f_n$</td>
<td>$c = f_c$</td>
<td>$c = f_n - f_c$ (capacities)</td>
</tr>
<tr>
<td><strong>approximate slope of total travel time function</strong></td>
<td>$a_1, a_2$</td>
<td>$a_1$</td>
<td>$a_2$ (cost coefficients)</td>
</tr>
</tbody>
</table>
in the arc pairs to obtain the link flows, and have a set of travel times which are equivalent to the travel time pattern related to the original problem.

In each arc pair, the arc which has the lesser cost coefficient $a_1$ will be called the "cheap arc" and will have an odd arc number. It may also be called the "odd arc". The arc with cost coefficient $a_2$ will be called the "costly", or "even arc". In Figure 2.2.2 D, we illustrated the network of arterial roads, specifying link numbers. Note that the link numbers are odd. From this point on, the diagram may be interpreted as representing arc pairs, in which the odd arc has the same number as its corresponding link, and the even arc has a number greater by one.

2.5.3 Additional properties of the arc-pair concept

Having presented the motivating argument for the introduction of the arc-pair scheme, we shall discuss some additional properties of the construction.

Let us consider the total time function for a link as derived from the two-piece linear approximation $T(f)$. Elementary calculus tells us that a piecewise linear function can be represented as the integral of a step function. If we call this step function $T^1(x)$ we can write

$$T(f) = \int_0^f T^1(x) \, dx = \begin{cases} a_1 f, & 0 \leq f \leq f_c \\ a_1 f_c + a_2 (f - f_c), & f_c \leq f \leq f_n \end{cases}$$

range 2

The function $T^1(x)$ defined by this integral can be interpreted as the increase in total travel time which results
from loading the x'th vehicle on the link. We can call $T_1$ the "incremental travel time function".

The function $T_1$ consists of two steps of respective heights $a_1$ and $a_2$, and widths $f_c$ and $(f_n - f_c)$. See Figure 2.5.3 A. We can interpret this figure in terms of our arc-pair construction. A link flow of value $f$ less than or equal to $f_c$ is loaded onto arc 1 entirely. The total travel time $T(f)$ is given by the hatched area, as in Figure 2.5.3 B. We can therefore interpret the hatched area in the first step of the step function to represent the total travel time $T_1$ spent on arc 1, and the shaded area in the second step to represent the total travel time $T_2$ spent on arc 2. See Figure 2.5.3 C. From previous arguments we see that the shaded area under the second step will be null unless the area in the first step is completely shaded.

The total travel time function for the odd arc is $T_1 = a_1 f_1$, where $a_1$ is the value of the link incremental travel time function $T_1$ for link flows less than or equal to $f_c$. The total travel time function for the even arc is $T = a_2 f_2$, where $a_2$ is the value of the link incremental time function $T$ for link flows between $f_c$ and $f_m$. The slopes $a_1$ and $a_2$ of the link travel time function $T(f)$ play the role of constant coefficients for the arc total travel times. The arc total travel times are therefore linear functions of the arc flows. Grouping in arcs into pairs corresponding to links will conceptually yield non-linear total travel time functions for the links but taking each arc separately gives us a set of linear total travel time functions.

The mathematical programming algorithm which minimizes the system travel time will not be required to group the arc flows into pairs. The program solves for the arc flows, and
FIGURE 2.5.3A
INCREMENTAL TRAVEL TIME FUNCTION AND THE ARC PAIR
FIGURE 2.5.3B  INCREMENTAL TRAVEL TIME: \( f < f_c \)
**FIGURE 2.5.3C**  INCREMENTAL TRAVEL TIME: $f > f_c$
the arc flows will be transformed to link flows after the minimization has been obtained. By this device we have "tricked" the mathematical program into "thinking" that its objective function is linear. The objective function is the sum over all arcs of the total travel time function for the arcs. Since this objective function is linear (and the other conditions will be shown linear) we have a linear program and can use the simplex algorithm to solve the problem.

The transformation from links to arcs is based on the approximation \( T(f) \) to the empirical total travel time function for the links \( T(f) \). Figure 2.5.1 A showed that this approximation lies quite close to the empirical function. Thus transforming the problem into a linear program costs almost nothing in terms of distortion of the cost function.

2.5.4 Algebraic form of the travel time functions

Next we sum up the previous notions, which were derived from graphical representations, by presenting the algebraic forms of the various functions.

(a) The empirical link average travel time function, \( t(f) \) has the form (See Figure 2.4.5 F.):

\[
\begin{align*}
 t(f) &= t_o + d_1 f & 0 \leq f \leq f_c \\
 t(f) &= t_c + d_2 (f-f_c) & f_c \leq f \leq f_n
\end{align*}
\]

In general, \( t \) is in minutes per mile per vehicle per lane, and \( f \) is in vehicles per hour per lane.

For an arterial street with 3 signals per mile, length one mile, one lane in each direction, loaded for one hour, with nominal speed limit 30 mph; we have:
t in minutes per vehicle
f in vehicles
\( t_c = 3.5 \text{ min/veh} \)
\( t_o = 3.0 \text{ min/veh} \)
\( t_m = 6.0 \text{ min/veh} \)
\( f_c = 500 \text{ veh} \)
\( f_n = 760 \text{ veh} \)
\( d_1 = 0.0010 = \frac{t_c}{f_c} \text{ min/veh}^2 \)
\( d_2 = 0.0150 = \frac{(t_n - t_c)/(f_n - f_c)}{(f_n - f_c)} \text{ min/veh}^2 \)

(b) The empirical link total time function has the form (See Figure 2.5.1 A):

\[ T(f) = f \cdot t(f). \]

(c) We have graphically approximated the function \( T(f) \) by a piecewise linear function, \( T(f) \) given by:

\[ T(f) = a_1 f \quad 0 \leq f \leq f_c \]
\[ T(f) = a_2 (f - f_c) + a_1 f_c \quad f_c < f \leq f_n \]

\( a_1 = 3.2 \text{ min/veh} \)
\( a_2 = 15.5 \text{ min/veh} \)

(d) We have shown \( T(f) \) to be the integral of a step function \( T_1(f) \) (See Figures 2.5.3 A, B, C.) given by:

\[ T_1(f) = a_1 \quad 0 \leq f \leq f_c \]
\[ T_1(f) = a_2 \quad f_c < f \leq f_n . \]

The units of \( T_1 \) are minutes per vehicle.
(e) From the approximate link total time function $T(f)$ we can derive a corresponding link average time function $t(f)$. This derived average travel time function agrees with the empirical average travel time function $t(f)$ which was the basis of our construction. The function $t(f)$ is plotted in Figure 2.5.4 A. We see that the agreement with $t(f)$ is close, as it should be since the approximate total time function was arranged to approximate the empirical data closely. Since the mathematical program calculates total travel times, Figure 2.5.4 A simply represents a check on the consistency of our approach.

$t(f)$ has the following form:

for $0 \leq f \leq f_c$:

$$t(f) = \frac{1}{f} \int_0^f T(x) \, dx$$

and for $f_c < f \leq f_n$:

$$t(f) = \frac{1}{f} \left[ \int_0^{f_c} T^1(x) \, dx + \int_{f_c}^f T^1(x) \, dx \right] =$$

$$= \left[ \frac{1}{f} \int_0^{f_c} a_1 \, dx + \int_{f_c}^f a_2 \, dx \right]$$
FIGURE 2.5.4A
COMPARISON OF EMPIRICAL AND DERIVED AVERAGE TRAVEL TIME FUNCTIONS
FIGURE 2.5.4B  DERIVED AVERAGE TRAVEL TIME FUNCTION $\bar{t}(f)$
\[ t(f) = \frac{1}{f} \left[ f_c a_1 + (f - f_c) a_2 \right] \]

This function is sketched schematically in Figure 2.5.4 B. Note from its algebraic form that it is asymptotic to \( a_2 \). This means that unlimited capacity on arc 2 would push the average travel time over both arcs (that is, over the link) to the average time characteristic of arc 2. Note that the constant portion of \( t(f) \) (for \( f \) less or equal to \( f_c \)) is the average of the empirical function \( t(f) \) for that range. That is:

\[ t(f) = \frac{(t_n - t_o)}{2} \text{ for } 0 = f = f_c. \]

It can be shown algebraically that the terminal value for \( t(f) \), that is, \( t(\hat{f}_n) \), equals the terminal value for the empirical function, \( t(f_n) \). This checks out because our piecewise linear total travel time function was fixed to intersect the empirical function at \( f = f_n \). (See Figure 2.5.1 A.)

We observe that the incremental time function \( T^1 \) for a link, which is not equal to the average travel time function for a link, has given us a pair of constant travel time coefficients (or cost coefficients) for the associated arc pair. Since the coefficient of travel time for an arc is constant, the incremental and the average travel time functions for the arc are both equal to the product of the arc flow and the arc cost coefficient. For a link, the incremental and average travel time functions are different; for an arc, they are equal.
See Figures 2.5.4 C, D, and E for a graphical representation of these relations among the derived travel time functions.

It can also be shown that \( t(f) \) is continuous at \( f_c \). However, \( t(0) \) is not equal to \( t(0) \). We recall that in constructing the approximation \( T(f) \) to the total link travel time \( T(f) \), we had \( T(f_n) \) about equal to \( T(f_n) \), \( T(f) \) continuous at \( f_c \), and \( T(0) \) equal to \( T(0) \). With the exception of the small discrepancy at zero flow we see that the average function \( t(f) \) derived from \( T(f) \) is as good an approximation to \( t(f) \) as \( T(f) \) is to \( T(f) \).

The consideration of the link travel time function has lead us to an approximation which allows us to formulate the objective function as a linear function. Assuming that the problem conditions can be expressed by linear equalities and inequalities, we therefore have a linear program.

We have shown how to define an arc capacity and a linear arc total travel time function in such a way that, by replacing each link with a pair of arcs, we simulate the original link capacity and the nonlinear link cost function.

In the next sections we shall present the mathematical formulations of the remaining problem conditions.
COMPARISON OF INCREMENTAL TRAVEL TIME FUNCTIONS FOR LINK AND ARCS
FIGURE 2.5.4D COMPARISON OF TOTAL TRAVEL TIME FUNCTIONS FOR LINK AND ARCS
FIGURE 2.5.4E COMPARISON OF AVERAGE TRAVEL TIME FUNCTIONS FOR LINK AND ARCS
2.6 The Matrix Representation of the Problem: The Network Configuration and the Commodity Input-Output Data

2.6.1 The node-arc incidence matrix

The configuration of nodes and arcs is represented by an array which we shall call $M$, the node-arc incidence matrix. The array has a row for each node in the network and a column for each arc. The matrix element $m_{ij}$ at the juncture of the $i$'th row and the $j$'th column is given by the rule:

$$m_{ij} = \begin{cases} 
1 & \text{if arc } j \text{ leads out of node } i \\
-1 & \text{" into "} \\
0 & \text{otherwise}
\end{cases}$$

A sample network is shown in Figure 2.6.1 A. The arc pairs corresponding to links are encircled by dotted lines. The corresponding node-arc incidence matrix is shown in the same figure. Note that the columns corresponding to the two arcs in a pair are identical, and that columns corresponding to opposing arcs are opposite in sign. Since an arc may only connect two nodes, there will be only two non-zero elements in each column.

The complete node-arc incidence matrix for the problem network shown in Figure 2.2.2 D is given in Figure 2.6.1 B.

2.6.2 Conservation of flow equations

We can use the node-arc incidence matrix to express the desire-line requirements. We have already remarked that the desire-line requirements can be restated as commodity flows on the network; for example, the flows from one or more origins to a destination can be considered a particular commodity flow type. In our basic problem we have three commodities. Satisfying the desire-line requirements is equiva-
FIGURE 2.6.1A  EXAMPLE NETWORK TO ILLUSTRATE NODE-ARC INCIDENCE MATRIX
FIGURE 2.6.1B  NODE-ARC INCIDENCE MATRIX M FOR PROBLEM NETWORK
lent to assuring the proper input of the related commodity at the origin nodes and the proper output at the destination nodes. Therefore for each commodity we have an input-output condition at every node:

\[
\text{(sum of flows on arcs leading out of the node) minus (sum of flows leading into the node) equals (commodity input or output to the network at the node).}
\]

Expressions of this form are referred to as conservation of flow equations, or Kirchoff node equations.

If the total flow of a commodity on arcs out of the node exceeds the total flow on arcs into the node, the node is an origin, or source for the commodity. The difference of these two sums, which is the net flow on arcs out of the node, equals the number of vehicles (of the given commodity type) which enter the network at that node. If the total flow on arcs out of the node is less than the total flow on arcs into the node, the net flow out of the node is negative. Then the node is a destination, or sink, for the commodity. The net flow in this case is a negative number; it is the negative of the number of vehicles (of the given commodity type) which leave the network at the node.

We next show how the node-arc incidence matrix can be used to express the conservation of flow equations.

Suppose we refer to the flow of commodity type k on arc j by the symbol \( f^k_j \). We can form the column vector of flows of commodity k, \( F^k \). We shall suppose these flow variables to be always non-negative. Physically this means that flow may not occur contrary to the arc orientation. The non-negativity is guaranteed in our computations by the simplex method.
The flow conservation equation then takes the form:

\[ \sum_i \left( f_i^k \right) + \sum_i \left( -f_i^k \right) = \text{(input of commodity } k \text{ at the node)}. \]

arcs leading into node
arcs leading out of node

where the symbol \( \sum_i \) denotes the operation of summing over index \( i \).

Referring to the definition of the node-arc incidence matrix \( M \), we see that a row of the matrix corresponding to one node contains a plus 1 in columns corresponding to arcs leading out of the node and a minus 1 in columns corresponding to arcs leading into the node. This is the same sign convention as that appearing in the flow conservation equation. Therefore, if we list the flow variables \( f_i^k \) in a column, so that the flows on the \( i \) arcs form the \( i \) components of the column vector \( F^k \), the scalar product of the \( i \)'th row of the node-arc incidence matrix \( M \) with the column vector \( F^k \) gives the net flow of commodity \( k \) out of node \( i \). That is,

\[ \sum_j \left( M_{ij} f_j^k \right) = r_i^k \]

where \( r_i^k \) if positive is the input to the network of commodity \( k \) at node \( i \), and if negative is the output from the system of commodity \( k \) at node \( i \). If we list these conservation equations for commodity \( k \), with one equation (row) for each node, we get the matrix equation:

\[ MF^k = R^k \]

\( R^k \) is a column vector with components \( r_i^k \); there is one component for each node. We call \( R^k \) the commodity input-output
vector, or the requirement vector. The above three equations have exactly the same content.

An example is given in Figure 2.6.2 A. Commodity 1 has an input of 10 units at node 1, an output of 5 units at node 2 and 5 units at node 3; commodity 2 has an input of 3 units at node 3 and an output of 3 units at node 1. Thus commodity 1 represents a flow with origin at node 1 and destinations at nodes 2 and 3; commodity 2 represents a flow with origin at node 2 and destination at node 1. The commodity input-output vectors are therefore

\[ r^1 = (10, -5, -5) \]
\[ r^2 = (-3, 3, 0), \]

where * denotes a transposed vector.

The conservation of flow equations for commodity 1 are:

for node 1: \[ +f^1_1 - f^1_2 - f^1_3 - f^1_4 + f^1_5 + f^1_6 - f^1_7 + 0 \cdot f^1_9 + 0 \cdot f^1_{10} = r^1_1 = 10 \]

for node 2: \[ -f^1_1 - f^1_2 + f^1_3 + f^1_4 + 0 \cdot f^1_5 + 0 \cdot f^1_6 + 0 \cdot f^1_7 + 0 \cdot f^1_8 - f^1_9 - f^1_{10} = r^1_2 = -5 \]

for node 3: \[ 0 \cdot f^1_1 + 0 \cdot f^1_2 + 0 \cdot f^1_3 + 0 \cdot f^1_4 - f^1_5 - f^1_6 + f^1_7 + f^1_8 + f^1_9 + f^1_{10} = r^1_3 = -5. \]

The equations for commodity 1 can be summed up as

\[ MF^1 = R^1 \]
OUTPUT 5 UNITS OF COMMODITY 1

INPUT 10 UNITS OF COMMODITY 1

OUTPUT 3 UNITS OF COMMODITY 2

INPUT 3 UNITS OF COMMODITY 2

OUTPUT 5 UNITS OF COMMODITY 1

FIGURE 2.6.2A ILLUSTRATION OF COMMODITY INPUT-OUTPUT REQUIREMENTS
with $M$ given in Figure 2.6.1 A, $R^1$ given above, and $F^1$ a vector of variables, representing flows of commodity 1 on the arcs.

The conservation of flow equations for commodity 2 are:

for node 1: $$-f_1^2 + f_2^2 - f_3^2 - f_4^2 + f_5^2 + f_6^2 - f_7^2 - f_8^2 + 0 \cdot f_9^2 + 0 \cdot f_{10}^2 = r_1^2 = -3$$

for node 2: $$-f_1^2 - f_2^2 + f_3^2 + f_4^2 + 0 \cdot f_5^2 + 0 \cdot f_6^2 + 0 \cdot f_7^2 + 0 \cdot f_{8}^2 - f_{9}^2 - f_{10}^2 = r_2^2 = +3$$

for node 3: $$0 \cdot f_1^2 + 0 \cdot f_2^2 + 0 \cdot f_3^2 + 0 \cdot f_4^2 - f_5^2 - f_6^2 + f_7^2 + f_8^2 + f_9^2 + f_{10}^2 = r_3^2 = 0$$

The equations for commodity can similarly be summed up as:

$$MF^2 = R^2.$$

We can combine the flow conservation equations for both commodities into one matrix equation by writing:

$$\begin{bmatrix} M & 0 \\ 0 & M \end{bmatrix} \begin{bmatrix} F^1 \\ F^2 \end{bmatrix} = \begin{bmatrix} R^1 \\ R^2 \end{bmatrix}$$

The matrix made up of the $M$ matrices has the so-called "block diagonal structure". The matrices form blocks along the diagonal; all elements outside these blocks are zero.
Were we to have more commodities on the same network we would continue to add the new commodity flow vectors in the column beneath the vectors $F_1^1$ and $F_2^1$. The new $R$ vectors would be listed beneath the present $R$ vectors, and the block diagonal structure of the new matrix would be extended.

We now have a means of completely specifying the network configuration and the desire line patterns in a set of equations which are clearly linear. The variables are the flows $F^k$; there is one flow vector for each commodity, and the components of a flow vector represent the flows of that commodity on each arc.

We remark that the node-arc incidence matrix need not be the same for all commodities; some commodities may be denied flow on particular arcs by removing these arcs from the incidence matrix related to that commodity.

At this point we have arrived at a matrix formulation for specifying network connectivity and the desire-line pattern. We must still provide matrix equations for the link capacities, the average link travel time function, and the system cost function. These provisions will follow easily from our previous analysis of the link travel time function.

2.6.3 Capacity constraints

We have shown how each link is replaced by a pair of arcs. The capacity of the odd numbered, or cheap arc, is $f_c$ and its cost per unit of flow $a_1$, the capacity of the second, or costly arc is $(f^n_c - f_c)$ and its cost per unit of flow is $a_2$. We can denote the capacity of the $j$'th arc by the symbol $c_j$. We require that the total flow on $j$'th arc be less than or equal to $c_j$, that is:
sum over all commodities (flow of k'th commodity on
arc j) ≤ c_j

or

\[ \sum_{k} f_j^k \leq c_j \]

The latter inequality, referred to as a "capacity restraint",
must be stated for each arc.

At this point we introduce a new set of variables,
which will be held non-negative by the linear program. They
will be called the "flow slacks", and will represent the un-
used flow capacity on the arcs. Denoting the flow slack on
arc j by s_j we can rewrite the capacity restraint inequality
as:

\[ \sum_{k} f_j^k + s_j = c_j \]

We can group these equations, one for each arc, into the fol-
lowing matrix equation:

\[
\begin{bmatrix}
I & I & \ldots & I
\end{bmatrix}
\begin{bmatrix}
F^1 \\
F^2 \\
\vdots \\
F^k \\
S
\end{bmatrix}
= C
\]

I denotes a unit matrix with one row for each arc;
C is a vector whose components are the arc capacities;
S is a vector whose components are the arc flow slacks.
2.6.4 **Average arc travel time and system time cost**

Our analysis of the average link travel time function led to a rule for setting up arc capacities (used in the capacity restraint equations) and the arc total travel time functions. The latter are just constant average travel time coefficients $a_1$ or $a_2$ (in general, $a_i$) times the total arc flows. These travel time functions need only appear in the expression for the system time cost, the objective function.

We next set up a matrix equation for the objective function. We have defined the objective function as the sum over all arcs of the total travel time on the arc. The objective function is

$$T = \sum \text{average travel time coefficient for arc} \times \text{total flow on arc}$$

The total flow on arc $i$ is the sum of all commodity flows:

$$\text{flow on arc } i = \sum_{k} f_{ik}^k$$

so that the objective function has the form:

$$T = \sum_{i} \left[ a_i \cdot \sum_{k} f_{ik}^k \right]$$

We interchange the order of summation to get

$$T = \sum_{k} \left[ \sum_{i} a_i f_{ik}^k \right]$$

which is the sum of total flow costs for all commodities.

Let $A$ be the row vector whose components are the average travel times for the arcs, $a_i$. The above expression for $T$ can be written, with the vector $A$ repeated once for each
commodity:

\[
T = \begin{bmatrix}
A, A, \ldots, A
\end{bmatrix}
\begin{bmatrix}
F^1 \\
F^2 \\
\vdots \\
F^k
\end{bmatrix}
\]

We are now in a position to group together the matrix equations for the input-output requirements, the capacity restraints, and the objective functions:

\[
\begin{bmatrix}
M \\
M \\
\vdots \\
M
\end{bmatrix}
\begin{bmatrix}
F^1 \\
F^2 \\
\vdots \\
F^k
\end{bmatrix} = \begin{bmatrix}
R^1 \\
R^2 \\
\vdots \\
R^k
\end{bmatrix}
\]

input-output

capacity restraint

(min) objective

This completes the mathematical expression of the basic problem: to find a set of routes and route loadings (in particular a set of commodity flows on arcs) which minimize the total travel time while satisfying the input-output requirements and the capacity restraints.

We have derived in great detail a model essentially equivalent to the Charnes-Cooper multi-copy traffic model. This model can be used to evaluate the effect on the total flow pattern and total travel time of any specified changes to the network. For example, average travel time coefficients may be modified, capacity restraints may be modified,
links may be added or deleted and so on. Parametric linear programming may be used to determine the sensitivity of the objective function to changes in problem parameters.

2.7 Extension of the Model to Problems of Optimal Network Design

As we pointed out in Chapter I, the formulation up to this point requires that the system analyst specify which network changes he wishes to investigate. From this shortcoming arises the computational difficulty of evaluating large numbers of alternative designs.

In the following sections, we extend the linear programming model to one which determines optimal variations in the network.

We shall set up three forms of this problem:

(a) Find the configuration of one way and two way streets and the resultant optimal flow pattern, which minimizes total travel time.

(b) Find the set of lanes which, when added to the network, yield the minimum total travel time (user cost) plus construction cost. That is viewing the set of users and the operators as one social entity, find the optimal joint plan for utilizing and improving the transportation system.

(c) Find the configuration of one way and two way streets and the minimum additions to road capacity (i.e., improvements to existing streets), and the resultant optimal flow pattern which has a
total travel time equal to or less than some required percentage of the travel time prior to improvements in the network (i.e., a specified improvement in travel time).

There is a conceptual device common to the solution technique for all three of the problems listed above. That device is the transformation of link capacity from a specified problem parameter, to a problem variable, to be determined through a mathematical programming algorithm. We shall not only be solving for an optimal set of flow values on the network, but simultaneously for an optimal set of capacities. The three problems stated above are variations on this theme. Each problem entails a characteristic set of constraints to be applied to the capacity variable, and a redefinition of the objective function.

The following section develops the mathematical formulation for each of these problems. The key to the formulation is the transformation of link capacity into a variable, but our previous discussion has shown that the mathematical presentation of the problem is ultimately in terms of arc capacities. We shall show how the transformation of arc capacities into problem variables can be done in such a way as to preserve approximately the shape of average travel time function for the link associated with an arc-pair.

2.7.1 The variable capacity formulation

We first simply rewrite the capacity restraint matrix equation
(I, I, ... I, I) \begin{bmatrix}
F^1 \\
F^2 \\
\vdots \\
F^k \\
S \\
C
\end{bmatrix} = C

as

(I, I, ... I, I, -I) \begin{bmatrix}
F^1 \\
F^2 \\
\vdots \\
F^k \\
S \\
C
\end{bmatrix} = 0

where \( \mathbf{0} \) is a column vector of zeros, one component for each arc.

Viewed simply as a transformation of the matrix equation, this operation adds nothing to the problem logic. But when the new form is incorporated into the mathematical programs, the vector of arc capacities \( \mathbf{C} \) is now treated as a variable:
If we write the transpose of the column of variables along the top of the constraint matrix, we have an alternative notion for the same mathematical program:

\[
\begin{bmatrix}
M & F^1 & R^1 \\
M & F^2 & R^2 \\
. & . & . \\
. & . & . \\
M & F^k & R^k \\
I & I & S \\
I & I & T \\
A & A & O \\
A & A & T
\end{bmatrix}
= 
\begin{bmatrix}
\end{bmatrix}
\text{input-output capacity restraint (min)}
\]

This representation can be interpreted in terms of the "activity analysis" terminology of linear programming (7). In the "activity analysis" interpretation, each variable in the linear program corresponds to an activity occurring in the system simulated by the linear program. This variable occurs at the head of a column whose coefficients are the inputs or outputs to the system of scarce resources, due to a unit level of operation of the activity. The j'th row of co-
efficient, in the constraint matrix, yields the total consumption of the j'th scarce resource for unit levels of all activities. Consequently the scalar product of the row of activity variables with the j'th row of coefficients yields the total consumption of the j'th scarce resource for arbitrary levels of the activity variables. Setting this scalar product equal to a given constraint is equivalent to specifying a material balance equation for the j'th scarce resource.

In our linear program the activities are \( F_1, F_2, \ldots, F_k \) and \( S \) and \( C \).

The j'th component of \( F_k, f_j^k \), corresponds to the activity of flowing \( f_j^k \) units of commodity \( k \) over arc \( j \). That is, \( f_j^k \) corresponds to the consumption of \( f_j^k \) units of capacity on arc \( j \). The j'th component of \( S, S_j \), corresponds to the activity of not utilizing \( S_j \) units of capacity on arc \( j \). The j'th component of \( C, C_j \), corresponds to the activity of supplying \( C_j \) units of capacity on arc \( j \):

\[
S) \sum_k f_j^k + S_j - C_j = 0
\]

This is a "material balance" equation for the j'th scarce resource, which is capacity on arc \( j \).

Dropping the slack \( S_j \) gives the equivalent inequality

\[
S) \sum_k f_j^k - C_j \leq 0
\]

or

\[
S) f_j^k \leq C_j
\]
Thus the "material balance" equation expressed in terms of capacity supply and consumption activities corresponds to our original notion of a capacity restraint on total flow in an arc. One might also consider this equation as representing a "conservation of capacity" condition, in analogy with the commodity input-output equations, which represent conservation of flow conditions.

Since the provision of capacity on an arc is now a variable in the mathematical program, we must recall at this point that the arcs in the network were an artificial construction intended to simulate the average travel time characteristics of the links in the actual network. To each link corresponded a pair of parallel arcs; for each of these arcs the average travel time and capacity were specified so that the behavior of flows on the arc pair, taken together, would represent the behavior of flows on the corresponding link.

It follows that the capacities of the arcs in a given pair must not be allowed to vary independently if the simulation of link behavior is to be maintained. We desire to express a relationship between the arc capacities (call them $c_a$ and $c_b$) which will have the following properties:

(a) The arc capacities, when set to unit levels $c_1$ and $c_2$, combined with the average travel times (call them $a_a$ and $a_b$) will generate the total travel time curve $T(f)$ for the original link. In other words, when $c_a = c_1$ and $c_b = c_2$, we have $T(f)$ for the original link, as in Figure 2.5.3 A.

(b) If the arc capacities $c_a$ and $c_b$ are set to integral multiples of the original values $c_1$ and $c_2$, (say
\[ c_a = n_1 c_1 \text{ and } c_b = n_2 c_2, \text{ } n_1 \text{ and } n_2 \text{ integers} \]

the effect is to simulate the additional of \( n \) links parallel to and identical to the original link.

(c) For any non-integral multiple of the original arc capacities, say \( c_a = p_1 c_1 \) and \( c_2 = p_2 c_2 \) (\( p_1 \) and \( p_2 \) not integers), the shape of the total travel time curve for the link implicitly simulated by the arc pair with capacities \( c_a \) and \( c_b \) will have essentially the same shape as the total travel time curve \( T(f) \) for the original link.

These conditions are satisfied if we require the arc capacities \( c_a \) and \( c_b \) to remain in a fixed proportion. Any increase in \( c_a \) entails a proportionate increase in \( c_b \). This is equivalent to requiring that \( n_1 = n_2 \) and \( p_1 = p_2 \) at all times. Then any change in link capacity can be specified by a single parameter, \( n_1 \) which multiplies the original arc capacities \( c_a \) and \( c_b \).

Figures 2.7.1 A, B, and C show the related effects of multiplying \( c_a \) and \( c_b \) by a parameter, on the incremental, total, and average travel time functions for the corresponding link. Multiplying the capacities \( c_a \) and \( c_b \) by a parameter (taken as 1.25 in the figures) shifts the functions to the right, while preserving the shape of the link total travel time function. When the parameter is not an integer, we call this process "fractional switching".

It is evident that multiplying both arc capacities by \( n \) is equivalent to rescaling the horizontal and vertical axes in these travel time diagrams, by a factor of \( n \). The effects of capacity restraints which were formerly observed at \( f = \frac{f}{c} \) and \( f = \frac{f}{n} \) are now observed at \( f = \frac{nf}{c} \) and \( f = \frac{nf}{n} \). This is equivalent to defining a new critical
FIGURE 2.7.1A  FRACTIONAL SWITCHING: INCREMENTAL TIME FUNCTION SHIFT
FIGURE 2.7.1B  FRACTIONAL SWITCHING: TOTAL TIME FUNCTION SHIFT
FIGURE 2.7.1C FRACTIONAL SWITCHING: AVERAGE TRAVEL TIME FUNCTION SHIFT
flow \( f_1^c = n f_c \) and a new maximal flow \( f_1^n = n f_n \). This is the physical significance of multiplying the arc capacities by a scale factor \( n \). If, for example, \( n = 2 \) we obviously represent a new link whose critical flow is twice the previous critical flow, and whose maximal flow is twice the previous maximal flow. Clearly our model represents two links where it previously represented one. This will be demonstrated formally below.

We shall next show that this scale factor can be interpreted as an activity variable: the activity consists of the decision to supply \( n \) arcs \( a \) and \( b \). To see this we simply rewrite the "conservation of capacity" equations for the two arcs corresponding to a link (call them again arc \( a \) and arc \( b \)):

\[
\begin{align*}
S) \quad & f_a^k + S_a - c_a = 0 \\
S) \quad & f_b^k + S_b - c_b = 0
\end{align*}
\]

Substituting \( c_a = n c_1 \) and \( c_b = n c_2 \):

\[
\begin{align*}
S) \quad & f_a^k + S_a - nc_1 = 0 \\
S) \quad & f_b^k + S_b - nc_2 = 0
\end{align*}
\]

Suppose the above equations correspond to arcs \( j_1 \) and \( j_2 \) respectively, where arcs \( j_1 \) and \( j_2 \) form the pair of arcs which simulate the \( i \)'th link.

Renaming subscripts we have:
Form a column vector $D$ whose $i$'th component corresponds to the activity of simultaneously supplying $d_1$ units of capacities $c_{j_1}$ and $c_{j_2}$ to the arcs $j_1$ and $j_2$ corresponding to the $i$'th link.

Then the above expressions can be interpreted as the scalar product of a new vector of activity variables ($F^{k*}$, $S^*$, $D^*$) with the rows $j_1$ and $j_2$ of a new constraint matrix.

In these terms our linear program takes the form

$$\begin{bmatrix} F^* & F^* & \cdots & F^* & S & D \end{bmatrix} \begin{bmatrix} R^1 \\ R^2 \\ \vdots \\ R^k \end{bmatrix} = \begin{bmatrix} 1 \\ I I \cdots I I \\ A A A \end{bmatrix}$$

input-output

capacity restraint

min

In the above tableau, and from this point on, the symbol $C$ denotes no longer the vector of arc capacities, but instead a capacity matrix which has one row for each arc in the network and one column for each arc pair (that is, for each link).

The matrix $C$ has the form:
It is clear that the $j_1$'st row of the new capacity restraint matrix equation corresponds to the equation

$$s_j^{k} + s_{j_1} - d_i c_{j_1} = 0.$$  

The linear program described by the above tableau constitutes the theme upon which we shall develop three variations in the following section. The properties which distinguish it from Charnes-Cooper multi-copy traffic network model are:

(a) the notion of variable capacity
(b) the introduction of a capacity supply decision variable

(c) the preservation of the shape of the link travel time functions by maintaining a constant proportion of capacity in the arc pair corresponding to each link

(d) effecting this constant proportion by multiplying the unit capacity of each arc in a pair (these unit capacities correspond to the simulation of a one-lane link) by a single capacity supply decision variable.

2.7.2 The one-way street model

Consider a rectangular grid of urban streets. Suppose the streets consist of one lane in each direction. (Neither the grid configuration nor the one lane assumption entails any loss of generality.) Given a set of flow input-output requirements for the network, we wish to orient the lanes in such a way as to permit an optimal flow at least total travel time.

The motivation for this problem arises from the fact that it costs virtually nothing to set up a system of one-way streets, so that finding an optimal one-way configuration appears to be a logical first step in planning improvements to an existing network.

Three factors influence the structure of the solution to this problem.

(a) Suppose we have a two-way street with one lane north and one lane south. The flow requirement pattern on the network is such that the dominant
flow is from north to south. Orienting both lanes southward primarily adds an extra lane of capacity to the southward direction, permitting more vehicles to flow along this road. The new maximal capacity becomes \( f_n = 2 f_n \).

(b) In the same example switching both lanes southward is equivalent to multiplying the capacities of the arcs representing the northbound lane (or link) by zero, and the capacities of the arcs representing the southbound arcs by two. In the previous section we pointed out that this is equivalent to rescaling the axis of the total travel time function for the southbound link. Whereas congestion effects were formerly observed at the critical flow \( f_c \), congestion effects are now not observed before the flow reaches the value \( f_c = 2f_c \). The second factor in the one-way problem is thus the ability to defer congestion effects to higher flows. In other words, one seeks to take advantage of the non-linearity of the travel time function.

(c) As the northbound lane is re-oriented in the southbound direction, the vehicles formerly flowing northbound in an optimal solution to the previously oriented network will have to seek other routes. This will lengthen the trip time for these vehicles and may add to the congestion on the new routes assigned to those vehicles. These negative effects must be weighed against the positive effect described in (a) and (b) in the determination of an optimum solution.
It is evident that a direct computation of these effects for various orientations in a network with flow requirements in many directions to say nothing of the large number of possible configurations, would be a lengthy and involved process. By formulating the problem as a mathematical program, we obtain an algorithm which leads to an optimal configuration, while avoiding the explicit enumeration of these effects.

To transform the general variable capacity program into a program for determining the optimal orientation of existing traffic lanes, we introduce two further conditions: the notion of fixed bundle capacity, and the integrality of the capacity supply decision variables.

We have previously introduced the relationship between a link and the corresponding arc pair, and we have interpreted varying the capacity of a lane as equivalent to varying the capacities of the related arcs in a fixed proportion. This is attained by multiplying both arc capacities by the same decision variable.

Next we consider the model as applied to a two-way street which connects two nodes. Suppose we represent each lane by a pair of arcs. The two pairs of arcs representing the two lane road will be called a "bundle". Suppose we wish to model a situation in which this two-way street may be left two-way, or set one-way in either direction. We approach this problem from the viewpoint of a continuous variation of the capacity of the link in each direction, subject to the restriction that the total capacity in both direction remains fixed. This means that any capacity removed from one direction must be added to the opposite direction, and conversely. This is the notion of fixed bundle capacity.
This reveals the motivation for our use of the expression "fractional switching" - this means the switching of a non-integral amount of capacity from one direction to the other.

For example, suppose the arcs corresponding to one link are numbered 1 and 2 and the arcs corresponding to the other link are numbered 3 and 4. Let the capacity supply decision variable for arcs 1 and 2 be named $d_1$, and for arcs 3 and 4 be named $d_3$. Then the capacity bounds on the flows take the form:

\[
\begin{align*}
S_1^k f_1^k + S_1 - d_1 c_1 &= 0 \\
S_2^k f_2^k + S_2 - d_1 c_2 &= 0 \\
S_3^k f_3^k + S_3 - d_2 c_1 &= 0 \\
S_4^k f_4^k + S_4 - d_2 c_2 &= 0 \\
\end{align*}
\]

These are the "conservation of capacity" equations.

The fixed bundle capacity constraint has the simple form:

\[
d_1 + d_2 = 2
\]

\[
d_1, d_2 \geq 0
\]

This condition states that one may supply between 0 and 2 times the original (unit) capacities $c_1$ and $c_2$ to arcs 1 and 2; the remaining fraction of total capacity must be assigned to the arcs in the opposite direction.
Suppose we set $d_1 = 1$. The decision variable $d_2 = 1$, and the "conservation of capacity" equations become:

\[
\begin{align*}
s_1^k + s_1 &= c_1 \\
s_2^k + s_2 &= c_2 \\
s_3^k + s_3 &= c_1 \\
s_4^k + s_4 &= c_2
\end{align*}
\]

Clearly this represents the original two way street with an arc pair of capacities $c_1$ and $c_2$ in each direction.

Setting $d_1 = 0$ gives the following "conservation of capacity" equation, with $d_3 = 2$:

\[
\begin{align*}
s_1^k + s_1 &= 0 \\
s_2^k + s_2 &= 0 \\
s_3^k + s_3 &= 2c_1 \\
s_4^k + s_4 &= 2c_2
\end{align*}
\]

These conditions state that the total flow on arc 1 and on arc 2 must be zero, while the total flow on arc 3 and 4 must be less than or equal to twice the original capacity of these arcs. Clearly this corresponds to replacing the original street by a one-way street with two lanes in the direction of the lane represented by arcs 3 and 4. Set-
ting \( d_1 = 2 \) corresponds to a one way street in the opposite direction.

We adopt the convention that the arcs in a bundle representing a two lane road shall be consecutively numbered, for example, \( i, i+1, i+2, i+3 \). The first and second arcs correspond to one link; the remaining arcs correspond to the opposite link. The integer \( i \) shall be odd. Arc \( i \) is the "cheap" arc corresponding to one link, arc \( i+1 \) is costly. Arcs \( i+2 \) (odd) and \( i+3 \) (even) play a similar role with respect to the opposite link.

By analogy with our "conservation of flow" and "conservation of capacity" equations, we can consider the condition

\[
d_1 + d_3 = 2
\]

a "conservation of capacity supply" or "conservation of allocation" condition for the one-way street problem. By virtue of this condition it suffices to specify the value of one decision (or switching) variable, \( d_1 \), for each bundle of arcs \( i, i+1, i+2, i+3 \), to determine the state of the bundle. We adopt the following terminology to describe the state of a bundle whose arc numbers begin with the integer \( i \):

(a) \( d_1 \) is any real number between 0 and 2:
the bundle is "free"

(b) \( d_1 = 2 \):
the bundle is "odd" (double capacity in direction of arc \( i \))

(c) \( d_1 = 0 \):
the bundle is "even" (zero capacity in direction of arc \( i \))
(d) \( d_1 = 1 \):

the bundle is "symmetric".

Having developed the notion of fixed bundle capacity and shown the significance of the integrality of the decision variable, we can now specify the complete mathematical program for determining the optimal configuration of one way streets. In matrix form this is:

\[
\begin{bmatrix}
  M & \mathbf{F}^1 \\
  M & \mathbf{F}^2 \\
  \vdots & \vdots \\
  M & \mathbf{F}^k \\
\end{bmatrix}
\begin{bmatrix}
  \mathbf{I}^1 \\
  \mathbf{S} \\
  \mathbf{D} \\
\end{bmatrix}
= \begin{bmatrix}
  \mathbf{R}^1 \\
  0 \\
  2 \\
\end{bmatrix}
\]

where \( \mathbf{I}^1 \) is a matrix with one row per bundle and two columns per bundle of the form:

\[
\mathbf{I}^1 = \begin{bmatrix}
  1 & 1 \\
  1 & 1 \\
  \vdots & \vdots \\
  1 & 1 \\
\end{bmatrix}
\]

and \( \mathbf{2} \) is a vector with one component per bundle, of value 2. \( \mathbf{D} \) is a vector with two components per bundle, namely \( d_i \) and \( d_i + 2 \), one decision variable for each arc pair in the bundle.
$I^1, D, \text{ and } 2$ generate the constraint:

$$I^1D = 2$$

or:

$$
\begin{bmatrix}
1 & 1 & & & \\
& 1 & 1 & & \\
& & 1 & 1 & \\
& & & & \\
& & & & & 1 & 1
\end{bmatrix}
\begin{bmatrix}
d_1 \\
d_3 \\
d_5 \\
d_7 \\
\vdots \\
d_b
\end{bmatrix}
= 
\begin{bmatrix}
2 \\
2 \\
2 \\
2 \\
\vdots \\
2
\end{bmatrix}
$$

or:

$$d_1 + d_3 = 2$$
$$d_5 + d_7 = 2$$
$$\vdots$$
$$d_{b-2} + d_b = 2$$

Thus $I^1D = 2$ represents the fixed bundle capacity condition, as claimed.

For a true one-way model we must further require that all components of $D$ be integer-valued. Therefore the mathematical program stated above falls in the class of "mixed-integer" programs. That is, the solution consists of a mixture of integer valued variables with variables not neces-
sarily integer valued. The latter variables are the $F^k$ and $S$. This is so because the flow variables will be large enough numerically so that rounding causes a very small percentage change. Rounding 998.2 vehicles to 998 is a good approximation. However, the decision variables lie between zero and two, any approximation results in a large percentage change. In physical terms, the decision to supply the capacity of 0.4 lane in one direction is quite different from the decision to supply zero capacity. (We shall in fact explore the effect of rounding decision variables.)

Because of the discontinuity of the integer valued variables a mixed integer program is fundamentally different from a linear program (the feasible region is not convex). However, it is possible to solve a mixed-integer program by solving a sequence of derived linear programs. The following chapter will discuss this method of solution and describe several numerical examples. Note that the decision variables $d_i$ were first introduced as continuous variables; their integrality was demanded subsequently. This approach corresponds to the logic of the algorithm which will be applied to solve the mixed integer program.

This completes the formulation of the problem of the optimal configuration of one-way streets as a mixed integer program. In the following section we describe our second network design problem.

2.7.3 The optimal link addition problem

This problem can readily be derived from the one-way street problem by reconsidering the "activity analysis" interpretation of the capacity supply decision variables.
The variables $d_i$ appeared in the "conservation of capacity" equations. For a given link, $d_i$ appeared in the equations for the corresponding arcs:

$$ \begin{align*}
S) & \quad f_i^k + S_i - d_i c_1 = 0 \\ 
S) & \quad f_{i+1}^k + S_{i+1} - d_i c_2 = 0
\end{align*} $$

Note that the cost coefficient for $d_i$ was set equal to zero. Furthermore, the "conservation of capacity" conditions related the allocation decision for a link to the allocation decision for a related link in the opposite direction.

Suppose, for this second problem, that the cost of allocating capacity is not zero, but can be measured according to a value scale which is common to the cost assigned to flow. One method for obtaining this is to assign a user cost in dollars for vehicles flowing on a road. The cost is based on fuel consumption, depreciation, insurance and time as a user costs. Further, we assign as capacity allocation cost the construction and maintenance costs of a link times the ratio of the time period during which average flow is simulated to the expected life-time of the road link. We assume that one can assign comparable costs $t_j^k$ to flows $f_j^k$, and $d_i$ to road capacity decision variables $d_i$.

Secondly, we delete the "conservation of allocation constraints", so that the decision variables are independent. We continue to require the integrality of the decision variables. These considerations lead to the following mathematical program:

...
where D is integral valued, and P is the vector of capacity allocation costs.

The above mathematical program thus represents the following problem: Given a network with certain roads existing or proposed, and given flow input-output requirements, determine a modified network in which lanes may be added to (or possibly deleted from) the original network, in such a way that user cost plus construction cost is minimized. The $d_i$ in this case represent the number of lanes to be supplied for link $i$.

If a given link $i$ is required to have a fixed number of links, say $n_i$, we may add a constraint

$$d_i = n_i$$

to the above programs, or we can delete the decision variable $d_i$ from the problem. In the latter case the conservation of flow equations

$$s_i^k + s_i - d_i c_i = 0$$
The matrix form for this problem will in general have two types of capacity equations: those for variable capacities and those for fixed capacities. The conservation of capacity matrix equation

\[
\begin{bmatrix}
I & I & \ldots & I & I & C
\end{bmatrix}
\begin{bmatrix}
F^1 \\
F^2 \\
\vdots \\
\vdots \\
F^k \\
S \\
D
\end{bmatrix}
= \begin{bmatrix}
0
\end{bmatrix}
\]

splits into a pair of matrix equations:

\[
\begin{bmatrix}
I_1 & I_1 & \ldots & I_1 & I_1 & 0
\end{bmatrix}
\begin{bmatrix}
F^1 \\
F^2 \\
\vdots \\
\vdots \\
F^k \\
S \\
D
\end{bmatrix}
= \begin{bmatrix}
0
\end{bmatrix}
\]
and
\[
\begin{bmatrix}
I_2 & I_2 & \cdots & I_2 & I_2 & C
\end{bmatrix}
\begin{bmatrix}
F^1 \\
F^2 \\
\vdots \\
F^k \\
S \\
D
\end{bmatrix}
= 
\begin{bmatrix}
C^1
\end{bmatrix}
\]

where \( D \) in this problem only appears for links of variable capacity. \( I_1 \) is a matrix whose \( i \)'th row adds the flow of a commodity on the \( i \)'th arc of fixed capacity, and \( I_2 \) acts similarly for arcs of variable capacity. The vector \( C^1 \) is the vector of capacities for arcs of fixed capacity. These equations can be gathered into the following matrix equation:

\[
\begin{bmatrix}
M \\
M \\
\vdots \\
I_1 & I_1 & \cdots & I_1 & I_1 \\
I_2 & I_2 & \cdots & I_2 & I_2 & C \\
T^1 & T^2 & \cdots & T^k & O & P
\end{bmatrix}
\begin{bmatrix}
F^1 \\
F^2 \\
\vdots \\
\vdots \\
F^k \\
S \\
D
\end{bmatrix}
= 
\begin{bmatrix}
C^1 \\
R^1 \\
R^2 \\
\vdots \\
R^k \\
0 \\
T
\end{bmatrix}
\]

\begin{align*}
&\text{conservation of flow} \\
&\text{conservation of fixed capacity} \\
&\text{conservation of variable capacity} \\
&\text{min cost}
\end{align*}

with \( D \) integral valued.

The solution of an example problem of this type via mixed integer programming will be discussed in the following chapter. The particular example is due to P. O. Roberts (8).
The example consists of a network on which certain links are proposed for construction. User costs and pro-rated construction costs are given, as are flow requirements. The problem is to determine the construction program (choose the links to be built) for minimum total cost. The problem differs from the above "general" formulation in two ways: some of the links are represented by only one arc (assumed absence of congestion effects for these arcs permits the assumption of constant travel time function over expected flow range) and a budget constraint limits the total construction expenditure.

2.7.4 The optimal cost improvement problem

We next proceed to our third network design problem by combining the link addition and one-way problems.

The optimal link addition problem can be combined with the one-way street problem in various ways. First, suppose we have a set of decision variables $d_i$ which have zero costs and are subject to "conservation of allocation" conditions. These are the "switching" variables for determining the one-way street pattern.

We can add to these variables another set of decision variables, $g_i$, which correspond to link addition variables.

The conservations of capacity equations for a bundle now have the form

\[
\sum_{k} f_k^{i} + S_i - d_i c_1 - g_i c_1 = 0
\]

\[
\sum_{k} f_k^{i+1} + S_{i+1} - d_i c_2 - g_i c_2 = 0
\]
These equations state that capacity may be allocated to each pair of arcs in a bundle via two decision variables. The first is the switching variable \(d_i\), which is subject to the fixed bundle capacity constraint but has zero cost; the second is the construction variable \(g_i\), which is so far unlimited, but has a positive cost attached to it.

The matrix form of the combined problem is

\[
\begin{bmatrix}
M & F^1 & R^1 \\
 & M & F^2 & R^2 \\
 & & \ddots & \ddots \\
 & & & M \\
I & I & \ldots & I & I & C & C & S & 0 & \text{conservation of flow} \\
 & & & & & I^1 & D & 2 & \text{conservation of capacity} \\
& & & & & & T^1 & T^2 & \ldots & T^k & 0 & 0 & P & G & \text{conservation of allocation} \\
& & & & & & & & & & & & T & \text{min cost}
\end{bmatrix}
\]

When \(G_1\) is the vector of construction decision variables, \(g_i\), \(D\) and \(G_1\) integral valued.

Suppose we relax the condition that the construction decision variables \(g_i\) be integral valued. In this case the model represents the possibility of fractional increases of
the capacity of a link. Construction is no longer restricted to building entire lanes. A fractional increase in the capacity of a link would correspond, for example, to widening a street, or improving the traffic signals, or forbidding parking. However, it makes no sense to forbid a fractional improvement in a direction opposite to that of a street which has been switched one-way. We can include this condition in a linear program as follows: if the switching variable $d_i$ for a given arc pair is zero, then no additional capacity $g_i$ can be added to this arc pair. We can require this by specifying that

$$g_i \leq nd_i \quad \text{or} \quad nd_i - g_i \geq 0$$

If $d_i$ is zero, then $g_i$ will be zero. If $d_i$ is positive, then $g_i$ may increase up to some multiple $nd_i$ of $d_i$, where $n$ is a parameter. The set of linear constraints of the above form is to be added to our mathematical program when we drop the condition that the $g_i$ be integral. We can call these constraints the "construction directionality conditions".

The program now has the form:

$$
\begin{bmatrix}
M & F^1 & R^1 \\
M & F^2 & R^2 \\
\vdots & \vdots & \vdots \\
M & F^k & R^k \\
I & I & \ldots & I & I & I & C & C \\
I & D & = & 2 \\
N - I & G & = & 0 \\
T^1 & T^2 & \ldots & T^k & 0 & 0 & P & = & T \\
\end{bmatrix}
\begin{bmatrix}
M \\
S \\
\vdots \\
F^k \\
S \\
D \\
G \\
P \\
\end{bmatrix}
= \begin{bmatrix}
R^1 \\
2 \\
\vdots \\
R^k \\
0 \\
2 \\
0 \\
0 \\
\end{bmatrix}
$$

conservation of flow

conservation of capacity

conservation of allocation

construction directionality

min cost
One more variation on this program will produce our final model. The reader may have been struck by a certain awkwardness in our assumption that the flow costs $T^k$ and the construction costs $P$ are commensurate. If a major component of flow cost is travel time, how are we to assign a reasonable average value of time to all users? To avoid this difficulty, we reformulate our problem in a way which appears natural in the context of the one way street problem. Suppose we have solved the original problem of finding the optimal configuration of one-way streets, with an optimal travel cost $T_o$. We now ask for the minimum additional construction necessary to permit an improvement of the travel cost $T$ by at least $f$ per cent beyond the optimum obtainable by switching alone.

To form a mathematical program from this formulation, we modify our combined switching and fractional addition program in the following manner.

The objective function was formerly the sum of flow costs plus construction costs. We now separate costs by writing the flow cost functional as a constraint:

$$S) T^k F^k \leq T_o - f T_o$$

The current objective function is simply the total construction cost:

$$P D = T$$

By this device we avoid the necessity of determining a common denominator for flow cost and construction cost.

The mathematical program becomes:
and \( D \) integral-valued.

The above program represents our final formulation of urban road network design problem in mathematical programming terms. Its solution specifies an optimal flow pattern, a simultaneous optimum switching pattern, and designates lanes whose capacity must be improved in a minimal cost improvement scheme.
2.8 Notes to Chapter II


(5) N. A. Irwin and H. G. von Cube, ibid.


(7) G. B. Dantzig, op. cit., Chapter 2.

CHAPTER III

Computational Experience with Mixed-Integer Mathematical Programming and Network Models

3.1 Introduction

In the previous chapter we developed a mixed-integer programming formulation for several types of network flow and synthesis problems. In this chapter we shall discuss a method of solution applicable to these problems, and we shall study in detail a set of numerical examples for the one-way street problem. We shall also solve an example problem of the link addition type. It will be seen that the optimal solutions to the example problems are by no means intuitively obvious. Moreover, a reasonable guess at an optimal solution can turn out to be in fact a very bad guess, when the guess is evaluated by mathematical programming. Under these conditions, the application of mathematical programming to this class of problems is necessary and justified.

The next section will be devoted to a discussion of the Land and Doig algorithm for mixed-integer programming. Following this we shall briefly describe the computational implementation of the algorithm. Then we present the results of our computational exploration of the optimal one-way street configuration for various flow patterns on a given network. A discussion of a link addition example problem concludes this chapter.
3.2 Solution of Mixed-Integer Programs

In the previous chapter we formulated a set of problems which would generate linear programs except for the requirement that certain variables take on only integer values in the problem solution. The variables which are required to be integer valued in the solution are the capacity allocation decision variables \( d_i \). The flow variables \( f_j \) need not be integral in a problem solution. Strictly speaking, they should be, for a non-integral number of vehicles is not physically meaningful. However, in dealing with large numbers of vehicles, rounding \( f_j^k \) to the nearest integer introduces only a small percentage error. Hence we accept non-integral \( f_j^k \) in a solution, rounding \( f_j^k \) to the nearest integer. The decision variables \( d_i \) are numbers between 0 and 2; clearly rounding any non-integral \( d_i \) may cause a large percentage error and does not produce an acceptable approximate solution.

In 1960 A. H. Land and A. G. Doig (1) presented a method for solving mixed-integer programs. The reader is referred to their paper for a complete discussion. We shall outline the essential points.

3.2.1 Lattice points in the feasible region

The problem is to minimize a linear objective function of \( n \) variables subject to a set of linear constraints, plus the additional condition that a set \( Q^l \) of \( q \) specified variables must take on integer values in the solution.

The space of primal variables, \( Q \), is \( n \) dimensional, and the linear constraints generate hyperplanes which bound a convex polyhedron \( R \). Within this \( n \) dimensional polyhedron,
the integrality condition restricts feasible solutions to
the lattice points in $Q^1$, a $q$ dimensional subspace of $Q$. That is, the feasible points are the intersections of $g$
hyperplanes each of which represents an integer value for
one of the $q$ variables in $Q^1$.

This is easily visualized in two dimensions. Suppose the primal space $Q$ is of dimensionality $n = 2$. The
linear constraints generate lines, bounding a convex polygon $R$. (See Figure 2.2.1 A.) Suppose both primal vari-
ables, $d_1$ and $d_2$, are required to take on integer values
($q = 2 = n$). Then the feasible points are the lattice
points within $R$, that is, the intersections within $R$ of
the hyperplanes (lines) for which $d_1$ and $d_2$ assume integer
values. This is a subset of the intersections of the lines
$d_1 = 0, 1, 2, 3 \ldots$ with the lines $d_2 = 0, 1, 2, 3 \ldots$.
Suppose that $d_1$ and not $d_2$ were required to assume integer
values. Than the feasible points are those on the lines
$d_1 = 0, 1, 2 \ldots$ and within $R$.

Our initial discussion will be in terms of a two di-
Mensional primal problem with both variables required to
assume integral values. The algorithm consists of an or-
dered search among the lattice points in the space $Q^1$ of in-
teger-valued variables. In an $n$ dimensional primal problem
with $(n - q)$ variables not required to be integral, we inves-
tigate the lattice points in $q$ dimensional space. The $(n - q)$
non-integral variables may take any values within the fea-
sible region $R$.

In our example, we have $n = 2$ and $q = 2$; there are
no non-integral variables. If on the other hand, we had
$n = 3$ and $q = 2$, we would still study the two dimensional
FIGURE 3.2.IA  POLYGON AND INCLUDED LATTICE POINTS
lattice (as shown in Figure 3.2.1 A) in the two dimensional subspace of integer variables $d_1$ and $d_2$. The convex set $R$ would be three dimensional; the $d_3$ axis would be perpendicular to the $(d_1, d_2)$ plane, and $d_3$ could assume continuous values in $R$. The point of this discussion is that one can ignore the behavior of the $n-q$ continuous variables when studying the lattice in $q$ dimensional space, to the extent that the continuous variables are assumed to remain feasible (this is checked in calculations).

The algorithm as applied to a two dimensional lattice with a convex polygon $R$, proceeds as follows. To begin, we suppose, contrary to the requirements of the problem, that $d_1$ and $d_2$ may take on not only integer values, but any values within $R$. Then as in an ordinary linear program, the minimal (or maximal) value of the objective function $T$ would lie at an extreme point of $R$. (See Figure 3.2.1 A) If this extreme point is not a lattice point of $Q^1$ this "optimal" solution is not a feasible solution to the integer program.

3.2.2 Searching among lattice points

In a minimizing problem, pushing the functional hyper-plane into the convex region increases its value, so that we wish to push it in as little as possible. Conversely, in a maximizing problem, pushing the functional down from an optimal extreme point into the convex region decreases the objective function. Hence we wish to push the objective function as little as possible into the polyhedron $R$. In this sense we wish to find the "first" lattice point encountered by the entering hyperplane. This is easy enough to do geometrically in two dimensions and possibly three dimensions, but how should one proceed in problems of higher dimensionality?
The search for the "first" lattice point, or the lattice point through which the objective functional hyperplane of greatest value (in a maximization problem) proceeds as follows.

Any lattice point is the intersection of \( q \) hyperplanes defined by integer values for \( q \) variables. We seek to construct this lattice point by choosing an integral value for one of the \( q \) variables. This defines a hyperplane. The other \( q-1 \) variables may take any values within \( \mathbb{R} \). Successively adding \( q-1 \) hyperplanes leads to the formation of a lattice point. The problem is, in which order does one select variables to be constrained to lie in integer hyperplanes, and what integer values shall be chosen to define these hyperplanes?

The following example and discussion will show that it is possible to order the selection of hyperplanes and to choose integer values in such a way that one maintains the highest possible value of the objective function throughout the process. When a lattice point is constructed, it will then have the highest possible objective value, and therefore will be an optimal solution to the integer program.

The mechanics of the search among lattice points are as follows. Setting \( d_i \) equal to some integer value \( j \) (forcing \( d_i \) to lie in the hyperplane \( d_i = j \)) is accomplished by adding a new linear constraint to the original constraint set which defined \( \mathbb{R} \). (This reduces from \( q \) to \( (q-1) \) the dimensionality of the space in which integer variables may still take continuous values. Within this \( (q-1) \) dimensional space we search for a new constraint to add. This is equivalent to searching for an integer value for some coordinate of the lattice point we desire to construct. When the dimension of the
search space has been reduced to zero, we have a lattice point. The difficulty lies, of course, in proceeding so as to construct the optimal lattice point.)

The objective function is maximized on the new convex domain $R_1^1$, (which is included in $R$) by solving a linear program to find that extreme point of $R_1^1$ which gives maximum $T$. This is the optimal objective value consistent with the current constraints. If this is a lattice point, the integer program is solved. If this extreme point is not a lattice point, another variable is required to be integer; that is, a new hyperplane is added to the constraint set and a new linear program is solved. (The rule for ordering this search has yet to be defined.)

3.2.3 A two-dimensional example

This procedure is illustrated by Land and Doig on a two-dimensional example, which we next present, adding our own illustrative diagrams. The objective function in this example is to be maximized, but the procedure for a minimization problem is analogous.

(a) Maximizing the objective function $T$ on the domain $R$ finds the extreme point $A = (4.2, 2.65)$ as "optimal" solution. (See Figure 3.2.3 A.)

(b) $A$ is not a lattice point. That is, it is not the case that the $d_1$ and $d_2$ coordinates of $A$ are both integral.

(c) Any solution to the integer program must have the $d_2$ coordinate integral. We add the constraint $d_2 = \text{integer}$. For a reason to be discussed below, we take the integer values for $d_2$ which were nearest the $d_2$ coordinate of $A$. These are (1) $d_2 = 2$ and (2) $d_2 = 3$. 
POINT A IS OPTIMAL WHEN INTEGRALITY CONSTRAINTS ARE IGNORED

FIGURE 3.2.3A INTEGER PROGRAMMING EXAMPLE
(1) Suppose we add the constraint \( d_2 = 2 \). (See Figure 3.2.3 B.) The feasible region \( R_1 \) is now of dimension 1 and is the line \( d_2 = 2 \), bounded by the faces of \( R \). Sliding the functional \( T \), out to its maximum \( T_1 \), finds an "optimum" at the extreme point \( B = (4.2, 2) \). But \( B \) is not a lattice point since \( d_1 \) is not integral.

(2) Suppose we add instead the constraint \( d_2 = 3 \). Maximizing \( T \) at \( T_2 \) finds the extreme point \( C = (2.35, 3) \). But \( C \) is not a lattice point since \( d_1 \) is not integral.

(d) The functional \( T_2 \) is greater than \( T_1 \). So we keep the constraint \( d_2 = 3 \) and next constrain \( d_1 \). The result will be a domain \( R_1 \) of dimension 0, a lattice point. At the optimum \( T_2 \), the value of \( d_1 \) was 2.6. We try the constraints \( d_1 = 3, \, d_1 = 2 \).

(1) Suppose we add the constraint \( d_1 = 3 \); the point \( (3,3) \) is outside \( R \), hence infeasible.

(2) Suppose we add the constraint \( d_1 = 2 \). The point \( D = (2, 2) \) is a lattice point and gives the optimal functional \( T_3 \) to the integer program. (See Figure 3.2.3 C.)

While this graphical illustration shows how one reduces the dimensionality of the space of variables by adding constraints, and that one finds a maximizing extreme point in this new space, iteratively until a lattice point is constructed, it does not show how one should pick the constraints, or why a given sequence of constraints leads to an optimum solution to the integer program.
POINT B IS OPTIMAL WHEN $d_2 = 2$
POINT C IS OPTIMAL WHEN $d_2 = 3$

FIGURE 3.2.3B  INTEGER PROGRAMMING EXAMPLE
Point D is the optimal lattice point.

**Figure 3.2.3c** Integer programming example
To see how one constructs the sequence of constraints, we look at another aspect of the problem.

3.2.4 Constructing the feasible domain of an integer variable as a function of the objective value

The objective functional hyperplane $T$ defines a subspace of the feasible domain $R$. That is, for any value of $T$, a set of points $d$ is defined which yield that objective function $T$. In particular, for any variable $d_i$, a minimum feasible value $d_{\text{a},i}$, a maximum feasible value $d_{\text{b},i}$ are defined for a given objective $T$. That is, given $T$, we can find a minimum $d_i$ and a maximum $d_i$ compatible with the constraints bounding $R$, and the given value of $T$.

For example, in Figure 3.2.4 A, the minimum and maximum values of $d_2$ given $T = T^n$ are $d_{\text{a},2}^n = 1.75$ and $d_{\text{b},2}^n = 3.2$. The integer points $d_2 = 2$ and $d_2 = 3$ are contained in between $d_{\text{a},2}^n$ and $d_{\text{b},2}^n$.

Note that this region is obtained by projecting the objective functional hyperplane on the $d_2$ axis. Land and Doig point out that by incorporating the objective function value as a variable in the original linear $n$ dimensional program, one obtains a new linear program in $n+1$ dimensions, corresponding to a new convex polyhedral set in $n+1$ dimensions. Projecting this set onto the $(d_i, T)$ plane yields a convex polygon. This polygon defines the feasible values of a given variable $d_i$ for all values of $T$.

In Figure 3.2.4 B we have illustrated the objective hyperplane $T$ falling across the feasible polyhedron down to its minimum. For each value of $T$, say $T_n$, the intersections of $T^n$ with the bounding hyperplanes are noted by the symbols...
FIGURE 3.2.4A  FEASIBLE RANGES OF $d_1$ AND $d_2$ FOR $T = T_n$
FIGURE 3.2.4B  FALLING OBJECTIVE PLANE AND FEASIBLE RANGES
The \( d_a \) coordinates of the points \( d_a^n \) and \( d_b^n \), that is, \( d_{b,1}^n \), denote the minimum and maximum feasible values, respectively of the variable \( d_1 \) for \( T = T^n \). The \( d_2 \) coordinates of the points \( d_a^n \) and \( d_b^n \), that is, \( d_{a,2}^n \) and \( d_{b,2}^n \) represent the minimum and maximum values respectively of the variable \( d_2 \) for \( T = T^n \).

In Figure 3.2.4 C we have constructed the \((d_2, T)\) plane, and we have plotted the points \( d_{a,2}^n \) and \( d_{b,2}^n \) for various \( T^n \). As expected the feasible domain for \( d_2 \) is a convex polygon.

The \((d_1, T)\) diagram, which we have constructed for our example problem, will form the basis for the derivation of general rules for the solution of the integer program.

Let us review the steps of our solution in terms of this diagram, which we shall call the "integer feasibility diagram".

Ignoring the integer constraints, we solved a linear program to find the extreme point \( A = d^o = (4.28, 2.60) \) and objective \( T^o = 4.27 \). Neither \( d_1 \) nor \( d_2 \) were integral. We next chose an integral value for \( d_2 \), \( d_2 = 2 \), and maximized the objective \( T \). This was a new linear program in a convex space of reduced dimensionality and the solution occurred at an extreme point. This extreme point was point B. The objective function was \( T^1 = 3.70 \). Similarly we tried \( d_2 = 3 \) and obtained the extreme point solution C, with \( T^2 = 4.04 \). This solution looks more promising, but \( d_1 = 2.7 \) is not integral. We added the constraint \( d_{1} = 2 \) and "optimized", although in this case there were no degrees of freedom left. The point D(2,3) is integral, and the objective \( T^3 \) = 3.80 was claimed optimal.
FIGURE 3.2.4C  INTEGER FEASIBILITY DIAGRAM
The procedure we followed yielded an integer solution but we still have not given the rationale for proceeding in this manner. What we are attempting to do is to search for an optimal lattice point by sequentially restricting ourselves to searching in subspaces of progressively smaller dimension. We need a rule for ordering this search, which amounts to determining an order in which we add integer constraints. Each successive constraint further restricts the search to a hyperplane in which a particular variable takes on only a known integer value. Further, we need to show why such an ordering leads to an optimum lattice point.

These arguments are obtained by considering the integer feasibility diagram from another point of view.

3.2.5 Obtaining an ordering rule by considering the convexity of the integer feasibility diagram

Ignoring integer conditions, we solved a linear program to find the optimum point $A$ at $T = T^0$. Consider lowering the functional hyperplane $T$ down across the integer feasibility diagram for the integer variable $d_2$, Figure 3.2.4 C. As $T$ slides down, the first time an integer point lies within the feasible region occurs when $T = T^2$. This integer point is $C$, with $d_2 = 3$. The value $T^2 = 4.2$ is an upper bound on solutions of the integer program, for there is no feasible solution (no integer value for $d_2$) for $T$ greater than $T^2$.

Defining $(d_i)$ to be the greatest integer contained in $d_i$, we next observe that $d_2$ at $C$, $d^2_b$, is $(d^o_2) + 1$. Since we know $d^o_2 = 2.6$ from the original linear program, we know that $(d^2_b) = 2$. The point $C$ corresponds to the maximum value of
the original linear program, given $b^2 = 2$. This remark is basic to the algorithm. It means that the point $C$ can be obtained by setting $d_2 = (d_2^0) + 1$ and maximizing $T$ by solving a linear program.

Suppose we continue dropping the functional hyperplane. The first new feasible integer value for $d_2$ occurs at $B$ when $T = T_1^1 = 3.7$ and $d_2 = 2$, but this is just $(d_2^0)$. So the point $B$ could be obtained by maximizing $T$ in a linear program, given $d_2 = 2$. Note that $T_1$ is another upper bound on solutions of the integer program, in the sense that there are no feasible solutions to the integer program when $d_2 = 2$ which have objective values greater than $T = T_2^2$.

These considerations lead us to a "branch and bound" procedure for solving the integer program. The solution can be represented as a tree diagram in which each node represents the maximization of the objective function subject to a set of integer conditions, by the solution of a linear program.

Figure 3.2.5 A shows the tree diagram for the example program. The first node represents the solution of an ordinary linear program, with objective $T = T^0 = 4.28$, and variables $d_1 = 4.15$, $d_2 = 2.6$. This corresponds to our solution point $A$. The variable $d_2$ must be integer valued in a solution, we then branch about the point $d_2 = 2.6$ to the points $d_2 = 2$ and $d_2 = 3$. Maximizing the objective function given these conditions gives points $B$ and $C$ with $T = T_1^1 = 3.7$ and $T = T_2^2 = 4.05$ respectively.

The previous discussion showed that the points $B$ and $C$ represent upper bounds on the objective function for the integer conditions $d_2 = 2$ and $d_2 = 3$ respectively. The
FIGURE 3.2.5A  TREE DIAGRAM FOR EXAMPLE PROBLEM
points B and C thus represent bounds on classes of solutions to the integer program.

We found, for the example problem, that the points B and C were not lattice points. For B, $d_1 = 4.25$ and for C $d_1 = 2.6$.

Our next step was to require $d_1 = 3$ in addition to $d_2 = 3$ and to maximize $T$. This gave $D_1$ with $T^3 = 3.8$. The solution $D$ satisfies the integrality conditions on $d_1$ and $d_2$ and is a candidate for an optimum solution. To prove that it is in fact optimum we must consider the bounding aspect.

So far, we know that there is no solution with $d_2 = 2$ and $T$ greater than 3.7, and there is no solution with $d_2 = 3$ and $T$ greater than 4.05. We have found a lattice point with $T = 3.8$. We must check whether there is any other lattice point with $T$ greater than 3.8. Clearly we can ignore any further points on the stem containing B, for the objective for any subsequent points on B is bounded from above by $T^1 = 3.7$. What about solutions in which we require $d_2 = 1$? Referring to Figure 3.2.4 C we see that the greatest objective $T$ for which $d_2 = 1$ is feasible is $T = 2.75$, quite far down.

In general, it is the convexity of the integer feasibility diagram which assures us that the further from the non integral point $d^0$ we go in requiring $d_1$ to be integral, the worse the objective must become. In other words, as we choose required values $d_1^n$ and find the maximum associated $T_1^n$, each $T^n$ serves as an upper bound to objective values for requirements further from the initial solution. We have already pointed out that each objective $T^n$ is an upper bound for all solutions which include the requirements on $d^n$ as a
subset of their requirements. Graphically, this is to say that the objective at each node in the tree diagram is an upper bound for all nodes further along the same stem.

These points lead us to conclude that we have both a horizontal and vertical bounding (or monotonicity) process in the tree diagram.

We use the term "horizontal bounding" in the following sense. We partition the tree diagram by assigning to each node a level: the nodes within a given level have the property that they all correspond to the same subset \( Q^n \) of integer variables each required to take some integer value. We name these levels \( L^n \): the \( n \)'th level collects all those nodes for which a given subset of exactly \( n \) variables have been required to take some integer value.

In Figure 3.2.5 A, \( L^0 \) contains only node \( A \), corresponding to the empty set of integer requirements. \( L^1 \) contains \( B \) and \( C \), corresponding to an integer requirement on variable \( d_1 \) only, and \( L^2 \) contains node \( d_1 \) corresponding to integer requirements on variables \( d_1 \) and \( d_2 \).

We define a "horizontal transition" in the tree diagram to be a transition from one node to another in the same level.

A proper vertical transition is one from a level \( L^j \) to a level \( L^{j+s} \) and \( s = 1 \); that is, exactly one variable \( d_i \) is added to the set \( Q^n \) to form \( Q^{n+s} \). A vertical transition is one for which \( s \) may be greater than 1; this corresponds to moving down along a sequence of proper vertical transition. Now we can define a proper horizontal transition as one in which that variable \( d_i \) moves from one required value \( d_i = k \) to another, \( d_i = k' \).

In Figure 3.2.5 A, some proper vertical and proper horizontal transitions are illustrated.
Now a variable $d_i$ which was added to the set $Q^n$ to form the set $Q^{n+1}$ had, in general, a non-integral value, $d_i = a$ in the solutions corresponding to the nodes in $L^n$. In performing a proper vertical transition, we require integer values $k$ for $d_i$ which carry $d_i$ a known distance, $p = (a - k)$, from the non-integral value $a$. In general, $d_i = a$ may be thought of as the point about which we are branching when we execute a proper vertical transition. We can now define an "increasing horizontal transition" as a proper horizontal transition not across the branch point value) in which $p$ increases, that is, the variable $d_i$ is moved further from its branch point value.

We can now restate the bounding conditions in terms of vertical transitions and increasing horizontal transitions:

For any vertical (resp., increasing horizontal) transition from node $A$ to node $B$, $T^A \preceq T^B$.

Thus we associate the "horizontal direction" with transitions across a given level. We can order the nodes in a given level according to increasing distance of the most recently added integer constrained variable, $d_i$ from its branch point value. The "horizontal bounding condition" states that associated with this ordering is an ordering of the objective functions associated with those nodes, such that the objective function will never increase under any (distance) increasing horizontal transition. In other words, the objective function at each node is an upper bound for the objective functions of all nodes reachable by increasing horizontal transitions from that node.
On the other hand, we associate the "vertical direction" with motion down along a stem of the tree, adding another variable to the set $Q$ of variables with specified integer values. An ordering and bounding process analogous to that for the horizontal direction is in effect.

Figure 3.2.5 B illustrates these relations for a set of nodes not related to our example.

In Figure 3.2.5 A, the vertical axis represents different values of the objective $T$. In Figure 3.2.5 B, this scaling is dropped and all nodes in a given level appear at the same height.

The computational advantage yielded by these bounding conditions is that they permit us to ignore all classes of solutions which have an upper bound worse than the upper bound for the node of current interest.

Returning to our example problem diagrammed in Figure 3.2.5 A we have the bounding conditions: $T^1$ at B bounds any objectives below B, and bounds the objective of F. $T^2$ at C bounds the objective at D and at E.

We have digressed from our analysis of Figure 3.2.5 A, where we had reached point D. At that point we had an integer solution with objective $T^4 = 3.8$. Our bounding conditions tell us that all solutions "beyond" point B (vertically and horizontally) have objective values not better than $T^1 = 3.7$, which is worse than our integer solution at D. However, the objective $T^2 = 4.05$ at C bounds any solution E. But this bound is greater than $T^4 = 3.8$. It is a priori possible that a horizontal transition from node C to node E might lead to an integer solution better than D.
PROPER VERTICAL TRANSITIONS:
\{AB, AC, BD, BE, BF, CG, CH, CI\} = P
VERTICAL TRANSITIONS:
P+ \{AD, AE, AF, AG, AH, AI\}

PROPER HORIZONTAL TRANSITIONS:
WITHIN \(l^2\), WITHIN \(l^2\), WITHIN \(l^1\)
HORIZONTAL TRANSITIONS:
WITHIN \(l^1\), WITHIN \(l^2\)
INCREASING HORIZONTAL TRANSITIONS:
ED, HI

FIGURE 3.2.5B  CLASSIFICATION OF TRANSITIONS
However, checking the integer feasibility diagram of Figure 3.2.4 C shows that no feasible solution exists for $d_2 = 4$. Therefore node $E$ and consequently the dependent node $H$ are not admissible solutions. From this we can conclude that there are no integer solutions with objective function greater than $T^4 = 3.8$. $T^4$ is the solution.

3.2.6 The general branch and bound algorithm

Figure 3.2.6 A is a generalized tree diagram representing the branch and bound algorithm. Each node represents the solution of a linear program, with certain variables required to be integral. Each node belongs to a level $L^i$, in which $i$ variables meet the integrality requirement. The goal is to move to a level $L^n$, in which all $n$ designated variables meet the integrality condition. The moves from $L^0$ to $L^n$ are to be carried out in such a way that the next move is always made from the node of highest objective function currently available. The bound conditions indicate directions of exploration which may be ignored until the current position is worse than some bound. At that time, exploration shifts to the region of the bounding node.

Let us trace the moves through Figure 3.2.6 A to illustrate these points. Node 1 corresponds to a solution in which integrality conditions are ignored - call this a "free" solution. A variable is selected, say $d_1$, and nodes 2 and 3 are calculated, branching to the nearest integers astride the free value for $d_1$. Nodes 2 and 3 belong to level $L^1$ and have objective $T^2$ and $T^3$. We next push $d_1$ one unit further from the branch point and check node 4 for a bound $T^6$. Of the nodes 2, 3, 4, node 2 has the greatest objective.
FIGURE 3.2.6A GENERALIZED BRANCH AND BOUND TREE
We choose a new variable, say $d_2$, and branch about its free value in node 2. This gives nodes 5 and 6. $T^5$ and $T^6$ are now compared with $T^4$. $T^4$ is a high bound, and the construction is continued from node 4.

While the argument of departing always from the node corresponding to the current upper bound is necessary to determine the optimum, in practice it may be economical to follow a chain completely down to an integral solution. This solution acts as a very strong bound on all other stems of the tree: any node whose objective function is worse than this bound immediately terminates any further searching along the stem containing that node.

We have found that a combination of the "highest node" method and the "complete stem" method has been computationally efficient. First, the highest node method is followed until variables begin to lean very strongly toward integer values. When this occurs, these variables are adjusted to their nearest integer values and one proceeds rapidly down a stem to obtain a complete integral solution and a bound. One must then check all other branches for bounds to be sure the current integer solution exceeds all other bounds. If not, the check continues.

Any node which corresponds to an infeasible solution similarly terminates a stem. In our problem, the integer variables are limited to values of $(0, 1)$ or $(0, 1, 2)$; in these problems it is unnecessary to calculate any nodes corresponding to integer values outside this range.

We have discussed this method of solution for a maximizing problem, but it is clear that analogous techniques apply to a minimizing problem. Instead of plotting a tree down from a free maximum solution, and seeking highest nodes, we
plot a tree up from a free minimum solution and seek lowest nodes.

This completes the description of the branch and bound algorithm for integer programming. We shall next describe the implementation of this algorithm on an electronic computer.

3.3 Computational Implementation

The foregoing discussion led to a characterization of the mixed integer programming algorithm as a sequence of linear programs, with integer requirements added. We showed that the requirements to be added depend on the non-integral value assumed by some variable in a previous solution, and that when certain variables assume almost integral values, it is appropriate to require these variables to assume those integral values.

These considerations suggest that it would be very useful to observe the progress of the algorithm, that is, the construction of the tree, and to have the facility of modifying the direction of the tree along which the algorithm is proceeding. It would also be convenient to be able to evaluate any proposed set of integer requirements - that is, any suggested capacity allocation decision.

We have been fortunate in having access to the MIT Project MAC Time-Shared IBM 7094 computer for our computation. This computer system (abbreviated as CTSS for "Compatible Time Sharing System") (2) permits direct communication with the computer via teletype, and affords the possibility of monitoring and supervising the algorithm.
Our computational system was therefore designed to take advantage of the MAC facility, and its particular features. We utilized as far as possible existing computer programs. This produced not the most efficient program, but one adequate for an experimental application of the algorithm.

We constructed our system as follows. We began with the linear programming code RSMFOR (3).

This is a very efficient all-in-core code. The code was modified for compatibility with the time sharing system in the following manner:

(a) a small time-sharing main program was written which set up an elapsed time indicator

(b) the RSMFOR main program BOS was recompiled as a subroutine

(c) the amount of matrix element storage required was reduced from 12000 to 7000 words (dimension of array A)

(d) the RSMFOR main program BOS was modified to clear storage not from a program breakpoint which it assumed available in location 99, but from an estimated breakpoint.

Absence of any intermediate scratch-tape usage by RSMFOR greatly simplified the job of conversion. We found, however, that the greatest time in executing the algorithm was consumed during the typing of output by the teletype. Accordingly we made the following input-output changes:

(a) Output routines were modified to give greatly reduced outputs. Subroutine SOT gave only the objective
value. Subroutine HOT gave only the name of a variable and its value in the solution.

(b) Statistics on basis inversion were suppressed.

(c) Output statements were changed from tape output to on-line (teletype) output.

After completing these modifications, we had a linear programming code available on the time-sharing system which would print out on the teletype the objective value (or an infeasibility message) followed by the variables in the order in which they were defined, and their values. Variables are defined in the linear programming code by the order in which their related columns are read in as input data. By reading in first those columns corresponding to decision variables, we obtained output priority for these decision variables. First the decision variables, then the flow variables would appear in the sequential output.

The time sharing system provides the facility of terminating a program at any time. By terminating after the values of the decision variables had been printed, we saved much output time. For a solution on which all integer requirements were met, we allowed the output to continue and obtained the flow solution.

Input to the linear program was organized as follows.

RSMFOR requires, basically, only definition of the objective function, a list of right hand side values for the linear equations (in which row name and value are specified), and a list of matrix entries in which column name, row name and value are specified. The matrix entries are read in by Subroutine TAP.
In order to execute the integer programming algorithm, we needed a way to enforce the integer requirements. The method we used was elimination from the problem of each variable set at an integral value. This elimination involved deletion of the corresponding columns, and the adjustment of the right hand side values to include the values of the integer variables.

The method most economical of computing time would be to carry out these eliminations within the matrix currently stored in core. Because of the involved storage techniques in RSMFOR, and because our program is experimental, we decided to sacrifice computer efficiency for the sake of economies in programming time. The method adopted was to read in the data matrix anew for each linear program with integer requirements (i.e., for each node on the tree) in such a way that a column corresponding to a required integer would be deleted, and the right hand side values adjusted for that integer value.

This was accomplished by modifying the matrix input routine, subroutine TAP. Before describing those modifications we point out that in the time-sharing system data is maintained on magnetic disk files. Editing or modifying a disk file is a program function whose execution time increases rapidly with the length of the file being edited. For reasons which will immediately be made evident, we split the data into two files, first a short file, then a long file. The short file, which can be edited quickly, contains the data most frequently revised.

Subroutine TAP was modified first to permit the reading of data which spread across two files - TAP recognizes a control statement which changes the input unit number. More
significantly, TAP is modified to recognize a special block of data at the head of the matrix data. This block contains a list of variable names which are to be set to integer values, and the corresponding values. Each record defined one integer variable and its value. In addition, provision was made for listing two variables in a record: the second variable was assigned a value (2-value of first variable). This was designed for convenience in solving the one-way street problem, in which there were two decision variables per bundle, and the sum of variable values was required to be 2.

After subroutine TAP absorbed the definitions of integer variables, it printed out this information, and went on to read the matrix element data. Each matrix element column name (variable name) was compared with all names in the list of integer variables. If there was no match, the matrix element was processed as usual. If there was a match, the element would not be stored, and the right hand side value for that row would be adjusted accordingly. A final check was made to see if these operations removed all matrix elements from any row. If so, the right hand side (having value zero) was deleted from the list of right hand side elements, and the "hole" left in the matrix data by the deletion of a row was closed up by relocating data.

After data was read in according to this scheme, the simplex algorithm was initiated.

Finally, a small control program was written to facilitate editing (that is, redefining integer values), checking current data values, and execution of the simplex algorithm.

We can summarize the technique by listing the operations involved in executing the algorithm.
(a) The revised linear programming code M4A is loaded on the disk. The objective function, right hand side data, and a few matrix element data are on a short disk file. The remainder of the matrix element data is on a long disk file.

(b) A command to CTSS will load and execute M4A, generating the linear programming optimum with no integer requirements. The output from this computation is an objective value and values for the decision variables $d_i$.

(c) A decision variable $d_j$ is chosen to be required integral. From this point on we use the control program. Calling the control program:

1. Puts the system in the edit mode. The short file is edited: The variable $d_j$ and its value are entered in the list of integer variables in the short file, and at the head of the matrix data.

2. Prints out the integer list as a check after the variables have been entered in the file.

3. Loads and executes M4A. M4A reloads the data, and TAP processes the integer information. The simplex algorithm minimizes the objective, subject to such integer requirements as have been defined.

The output from the integer program may be terminated by a break signal to the CTSS supervisor program.

The output is used to plot a point on the tree diagram. A decision is made as to the next integer requirements to be investigated (i.e., the next node to be plotted). The control program is then re-started for another step of the calculations.

This concludes the description of the computational facilities for solved mixed-integer programming problems with
the branch and bound technique. We remark that in a sense we apply the algorithm manually: decisions as to transitions are made by the analyst. We use the computer as a tool for rapid execution of the simplex algorithm and for convenient manipulation of the integer requirement information.

In the next section we begin our description of the application of this technique to sample problems.

3.4 The One-Way Street Problem with Three Commodities

In the previous chapter we derived the mathematical model for analyzing the traffic flow on a network of arterial streets. The network which serves as a basis for our numerical examples has the following characteristics.

There are nine nodes, which are connected by a rectangular grid of arterial links. Each link has a length of one mile, with an average of three traffic signals per mile. There are a total of twelve links. Each link has nominally one lane in each direction. Each lane is represented by a pair of arcs. These arcs are numbered consecutively from 1 through 48. Lane i is represented by the arcs numbered i and i+1, where i is odd. The first, or odd numbered arc, is "cheap", the even numbered lane is "costly".

Figure 2.2.2F shows the odd numbered arcs. The even numbered arc should be visualized as implicitly parallel; it is a parasite arc in the sense that any change of the capacity of the odd arc affects the even arc identically. In the previous chapter we analyzed the average travel time function and derived the following nominal parameters, which we assign equally to all lanes.
odd arc: capacity 500 vehicles (per hour) = $c_1$
average cost: 3.2 minutes per vehicle
= $t_1$

even arc: capacity 260 vehicles (per hour) = $c_2$
average cost: 15.5 minutes per vehicle
= $t_2$

Using the node-arc incidence matrix $M$ for this network, and using the above cost and capacity values, we have a mathematical program of the form written down in the section containing the formulation of the one-way street problem, lacking only specification of the commodity inputs and outputs.

3.4.1 Loading levels and loading patterns

In the computations to be described in this section, we shall keep the network, capacities, and costs fixed, except for possible reallocation of capacity from one lane to another within a link. We shall call this capacity orientation process the process of "switching".

In exploring the types of solutions to be obtained for this mathematical program, we shall vary the input-output requirements (or loadings) of the network, in order to learn how the one-way street (i.e., "switching") pattern depends on loading factors.

We break down the notion of loading factors into two aspects: the aspect of pattern and the aspect of level.

By pattern, we mean the relative dominance of one or several flow directions. If a dominant flow direction is discernible one would expect the one-way streets to favor that direction as being the most demanding of capacity.
By level, we mean the average ratio of lane flow to lane capacity.

The input-output requirements we shall use for these examples are based on the notion of "radial and circumferential flows" discussed in the previous chapter. The "radial flow" (commodity 1) enters the network at nodes 1 and 3, and leaves the network at node 8. The first circumferential flow (commodity 2) enters the network at node 7 and leaves at node 3. The second circumferential flow (commodity 3) enters the network at node 9 and leaves at node 1. See Figures 2.2.2 C and F.

We organized our search of the solution space as follows. First, we decided on three basic pattern configurations. In the first pattern, the load was the same at each input point. That is, the flows from node 1 to node 9, node 3 to node 8, node 2 to node 7, and node 8 to node 3 all value the same value.

This produces a "crisscross pattern" of flows on the grid network in which it is not at all evident which is the dominant flow direction. We call this an isotropic pattern. The aim was to choose a pattern for which one-way solution would not be evident. After exploring the properties of the isotropic pattern, we changed the relative intensities of the commodity flows to produce "biased" patterns. As expected, the character of the switching solution changed markedly if the bias was large. In the latter case we call the pattern "lopsided".

We can summarize the organization of the computational examples for the one-way street problem as follows. Three types of pattern were studied: the isotropic, the biased,
and the lopsided. These terms refer to the character of the input-output requirement structure of a given loading. An isotropic, biased or lopsided pattern of loading is given as input to the integer program. The program then determines values of flows and the values of switching decision variables. A "solution" to the integer program consists of an optimal basis to the set of constraints, together with values for the flow and switching variables, and an objective function value. When we say that a certain solution is optimal for a set of different input-output requirements (that is, various levels and/or various patterns), we mean that the same basis is optimal for all problems in the set. This means that the switching variables will have the same values for all problems in the set, but that the values of the flow variables in the basis will be adjusted to match the current input-output requirements. In other words, the switching pattern remains the same, and so does the pattern of non-zero flow variables, but the flow variables may assume different non-zero values.

The examples were classed according to the three loading patterns: isotropic, biased, and lopsided. For each of these patterns the range of feasible solutions was investigated for loading levels from zero to 2000 vehicles (a load well above capacity) loaded at each input point. For each loading pattern, this ranging was carried out independently for three policies with respect to the switching variables.

The first policy, which we call the "suggested network", assigned values to switching variables which corresponded to a one-way pattern which was thought, as a guess, to represent a good configuration. The second policy, which we call the symmetric network, assigned values to decision variables
which corresponded to all streets being two-way. The terminology "symmetric" is taken from graph theory where a graph is termed symmetric if, for every oriented arc, there exists an oriented arc in the opposite direction. The third policy consists in allowing the decision variables to take any (continuous) values between zero and two. This corresponds to the initial step of the branch and bound algorithm. Tables 3.4.3 (1) and Figure 3.4.2 B show the results of this investigation for the isotropic loading pattern.

The effects of the various switching policies on the maximal feasible load and on the network flow costs are demonstrated by that data.

We can schematize the structure of this computational approach by the following tableau:

<table>
<thead>
<tr>
<th></th>
<th>suggested network</th>
<th>symmetric network</th>
<th>free network</th>
</tr>
</thead>
<tbody>
<tr>
<td>isotropic load</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>biased load</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>lopsided load</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
</tbody>
</table>

where L denotes the solution of a sequence of linear program over the range of feasible load levels.

The next phase of computation began with the selection of a suitable load level for the execution of the branch and bound algorithm. Having investigated the range of feasible solutions, we selected load level 950 (which is the feasibility limit for the symmetric network with isotropic load) as the level for which we desire to determine the optimal switching pattern. For each loading pattern, isotropic, biased,
and lopsided, we executed the branch and bound algorithm for load level 950.

This phase of computation can be schematized by the following tableau:

<table>
<thead>
<tr>
<th>optimal integer-switching network</th>
</tr>
</thead>
<tbody>
<tr>
<td>isotropic load</td>
</tr>
<tr>
<td>biased load</td>
</tr>
<tr>
<td>lopsided load</td>
</tr>
</tbody>
</table>

where L 950 denotes the execution of the branch and bound algorithm for load level 950.

The branch and bound algorithm is a general approach to the problem of integer programming on networks. In examining the cases listed above, certain special situations were observed. A complete discussion of the results of computation is given in the following three sections which correspond to the three load patterns.

By way of introduction, we shall discuss these special situations (which, although special, are not rare) in general linear programming terms. These situations will be discussed in terms of the particular examples in the sequel.

The first special situation is the existence of multiple optimal solutions to a program. In such a case, there is more than one set of values of flows and switching values which produce the same objective function value. In terms of the geometry of convex sets, this occurs when an optimal extreme point lies in a face of the convex polyhedron which is parallel to the objective hyperplane.
In terms of the network flow model, this means that there are several configurations of paths along which commodities flow which are equally satisfactory as to total time cost. It may also mean that several switching configurations are equally good.

It is reasonable that problems on our network possess multiple optima. All arcs have the same length; this permits the construction of various paths of the same cost. We found multiple optima for the isotropic and biased load examples; there may be multiple optima for the lopsided load, but we did not compute them.

The second special situation encountered in the integer program is the existence of an integer solution with the same objective function value as the optimal linear programming solution when integer conditions are ignored (the "free solution"). This occurs when the optimal extreme point for the linear program has integer-valued coordinates (i.e., is a lattice point), or, multiple solutions to the linear program existing, there is a lattice point lying in that bounding hyperplane which contains the optimal extreme point. (The lattice point may or may not be an extreme point).

This situation occurred in the solution of the integer program for the isotropic case. The initial (free) solution had non-integral values for the decision variables, and we constructed two integer solutions with the same objective value. These solutions lie in the same bounding hyperplane. One does not attempt to determine this by constructing the n dimensional polyhedron, but by solving the integer program.
The third special situation is the validity of a basis as an optimal solution for a range of requirement values (right-hand-sides of the linear program). In the switching problem we determined that two of the four optimal bases for the biased loading were also optimal for the isotropic loading. (The remaining two optimal bases could also be checked). This means that the switching and flow patterns were the same in these optimal solutions for the isotropic and the biased loads. Flow values and the objective value would, of course, be adjusted to the new requirement vector.

The fourth special situation is the ability to jump to a good integer solution fairly early in the branch and bound algorithm. In the network problems we have investigated, one cannot, in general, round the decision variables obtained in the "free" solution to the nearest integer. After constraining one-third to one-half of the switching variables to integer values, we found that the remaining decision variables would lie very close to integer values. They could then be rounded to integer values to obtain a good solution.

Physically, this corresponds to the fact that after some of the switching variables are set to "good" values as indicated by the rules of the branch and bound algorithm, the conservation of flow conditions will demand capacity in particular directions. This condition is manifested by the switching variables assuming nearly integer values. For the lopsided loading, it was possible to jump very quickly to an all integer (decision variable) solution. Finally, we point out that the computations showed that a heavy penalty is attached to rerouting traffic, due both to the in-
creased demands on network capacity. In general, one must solve the integer program to determine whether this penalty is offset by the gains obtained from the one-way system. This suggests a conservative heuristic: in designing a one-way pattern, minimize the rerouting of traffic. If one sets those streets one-way which had zero or negligible flow in the blocked direction, rerouting will be small or negligible. Such a solution may yield small gains while not being optimal, or it may represent the best one can do if there the flow demands on the network are heavy and there is little room for adjustment - little free play in the system. In the isotropic and biased load examples, such a conservative solution did in fact belong to the set of optimal solutions.

In this section we have tried to point out that certain general linear programming principles underlie the relationships of the various solutions. Exactly when a given linear programming situation hold depends on the input to a given linear program. We have common phenomena in our examples because they are all linear programs; these phenomena are significant or not depending on the data of the problem. We have tried to bridge the gap between a general, theoretical interpretation of these phenomena, and a physical interpretation in terms of flows on networks.

In the following three sections, these factors will be discussed in terms of the details of the computational examples.

3.4.2 The isotropic pattern

We studied the isotropic pattern first to determine the range of feasible loadings and the associated costs. We ranged the loadings for all commodities simultaneously
keeping their ratios, (i.e., the pattern) fixed from very small input-output requirements (low levels) to very large requirements (high levels).

This was done by solving a linear program for each loading level, given conditions on the decision variables. (It would have been more efficient computationally to use a parametric programming algorithm which permits automatic scaling of the right hand side, but the RSMFOR code, unlike other slower but more elaborate codes, does not possess this feature.) The load ranging was done for two conditions on decision variables: (a) the decision variables were set to assign unit capacities $c_1$ and $c_2$ in each direction - that is, all streets were two-way (symmetric); (b) the decision variables were allowed to range freely within the bundle capacity constraints. In the latter case the capacities would be automatically aligned in the direction of greatest utility by the mathematical program. We call this the "free network".

Table 3.4.2 (1) shows the results of ranging the isotropic load on the symmetric network and on the free network. The levels listed correspond to the load entered at each of the 4 entry points, in vehicles, during a one hour loading period.

The total load is four times the level value, given in thousands of vehicles (kilovehicles).

The costs correspond to total travel time spent by all loaded vehicles, in thousands of minutes (kilominutes).

Table 3.4.2 (1) also shows a cost column for a "suggested" network. This network consists of a street orientation pattern which seemed by inspection to be a reasonable solution to the optimum orientation problem. The suggested net-
work is deputed in Figure 3.4.2 A. Note that it is based on setting up feasible circulatory flows within the network. This is a method frequently employed by practicing traffic engineers. We shall see how it compares with the optimum solution.

Table 3.4.2 (1) and its graph, Figure 3.4.2 B, tell us a great deal about what can be gained by reallocating capacities in a network.

Curve B represents the network with all streets in a two-way (symmetric) state. This might be considered the natural, or initial state of the network. As the load is increased, the total cost increases linearly until the load passes 2500 vehicles. This is slightly above the 600 vehicle input level. From that point on costs rise at an increasing rate until the maximum feasible input is reached at a level of about 950 vehicles (the actual cut-off point lies somewhere between the 950 and 960 vehicle levels).

One major point about this graph is that the total cost curve, as a function of total load, for the entire network, has essentially the same shape as the corresponding curve for a single lane (arc-pair). With a relatively isotropic flow pattern, one can legitimately think of the congestion on a network as having the same form as congestion on a lane.

The second major point is that there is somewhat more leeway in the network than in the lane for adjustment to increased demand. The rapid increase in total cost begins in a lane at the critical flow (the saturation of the "cheap arc"). This point is at a lane flow of 500 vehicles. The
FIGURE 3.4.2A  SUGGESTED NETWORK ORIENTATION FOR ISOTROPIC LOADING
### TABLE 3.4.3 (1)

Total Costs for Ranged Isotropic Loads for Various Network Orientation Patterns

<table>
<thead>
<tr>
<th>Level (veh.)</th>
<th>Total Load (kiloveh.)</th>
<th>A</th>
<th>Suggested Network Cost (kilomin.)</th>
<th>B</th>
<th>Symmetric Network Cost (kilomin.)</th>
<th>C</th>
<th>Free Network Cost (kilomin.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.400</td>
<td>4.479</td>
<td>4.479</td>
<td>4.479</td>
<td>4.479</td>
<td>4.479</td>
<td>4.479</td>
</tr>
<tr>
<td>200</td>
<td>0.800</td>
<td>8.959</td>
<td>8.959</td>
<td>8.959</td>
<td>8.959</td>
<td>8.959</td>
<td>8.959</td>
</tr>
<tr>
<td>400</td>
<td>1.600</td>
<td>17.919</td>
<td>17.919</td>
<td>17.919</td>
<td>17.919</td>
<td>17.919</td>
<td>17.919</td>
</tr>
<tr>
<td>500</td>
<td>2.000</td>
<td>22.399</td>
<td>22.399</td>
<td>22.399</td>
<td>22.399</td>
<td>22.399</td>
<td>22.399</td>
</tr>
<tr>
<td>700</td>
<td>2.800</td>
<td>31.359</td>
<td>35.049</td>
<td>35.049</td>
<td>31.359</td>
<td>31.359</td>
<td>31.359</td>
</tr>
<tr>
<td>800</td>
<td>3.200</td>
<td>infeasible</td>
<td>46.909</td>
<td>46.909</td>
<td>40.759</td>
<td>40.759</td>
<td>40.759</td>
</tr>
<tr>
<td>900</td>
<td>3.600</td>
<td>62.705</td>
<td>62.705</td>
<td>62.705</td>
<td>55.079</td>
<td>55.079</td>
<td>55.079</td>
</tr>
<tr>
<td>910</td>
<td>3.640</td>
<td>64.875</td>
<td>64.875</td>
<td>64.875</td>
<td>56.511</td>
<td>56.511</td>
<td>56.511</td>
</tr>
<tr>
<td>920</td>
<td>3.680</td>
<td>67.045</td>
<td>67.045</td>
<td>67.045</td>
<td>57.943</td>
<td>57.943</td>
<td>57.943</td>
</tr>
<tr>
<td>930</td>
<td>3.720</td>
<td>69.215</td>
<td>69.215</td>
<td>69.215</td>
<td>59.375</td>
<td>59.375</td>
<td>59.375</td>
</tr>
<tr>
<td>940</td>
<td>3.760</td>
<td>71.385</td>
<td>71.385</td>
<td>71.385</td>
<td>60.807</td>
<td>60.807</td>
<td>60.807</td>
</tr>
<tr>
<td>950</td>
<td>3.800</td>
<td>73.555</td>
<td>73.555</td>
<td>73.555</td>
<td>62.239</td>
<td>62.239</td>
<td>62.239</td>
</tr>
<tr>
<td>960</td>
<td>3.840</td>
<td>infeasible</td>
<td>63.671</td>
<td>63.671</td>
<td>65.103</td>
<td>65.103</td>
<td>65.103</td>
</tr>
<tr>
<td>970</td>
<td>3.880</td>
<td>65.535</td>
<td>65.535</td>
<td>65.535</td>
<td>67.967</td>
<td>67.967</td>
<td>67.967</td>
</tr>
<tr>
<td>1000</td>
<td>4.000</td>
<td>69.399</td>
<td>69.399</td>
<td>69.399</td>
<td>69.399</td>
<td>69.399</td>
<td>69.399</td>
</tr>
<tr>
<td>1010</td>
<td>4.040</td>
<td>71.569</td>
<td>71.569</td>
<td>71.569</td>
<td>71.569</td>
<td>71.569</td>
<td>71.569</td>
</tr>
<tr>
<td>1020</td>
<td>4.080</td>
<td>73.739</td>
<td>73.739</td>
<td>73.739</td>
<td>73.739</td>
<td>73.739</td>
<td>73.739</td>
</tr>
<tr>
<td>1030</td>
<td>4.120</td>
<td>75.909</td>
<td>75.909</td>
<td>75.909</td>
<td>75.909</td>
<td>75.909</td>
<td>75.909</td>
</tr>
<tr>
<td>1040</td>
<td>4.160</td>
<td>78.079</td>
<td>78.079</td>
<td>78.079</td>
<td>78.079</td>
<td>78.079</td>
<td>78.079</td>
</tr>
<tr>
<td>1050</td>
<td>4.200</td>
<td>80.249</td>
<td>80.249</td>
<td>80.249</td>
<td>80.249</td>
<td>80.249</td>
<td>80.249</td>
</tr>
<tr>
<td>1060</td>
<td>4.240</td>
<td>82.419</td>
<td>82.419</td>
<td>82.419</td>
<td>82.419</td>
<td>82.419</td>
<td>82.419</td>
</tr>
<tr>
<td>1070</td>
<td>4.280</td>
<td>84.589</td>
<td>84.589</td>
<td>84.589</td>
<td>84.589</td>
<td>84.589</td>
<td>84.589</td>
</tr>
<tr>
<td>1080</td>
<td>4.320</td>
<td>86.759</td>
<td>86.759</td>
<td>86.759</td>
<td>86.759</td>
<td>86.759</td>
<td>86.759</td>
</tr>
<tr>
<td>1090</td>
<td>4.360</td>
<td>88.929</td>
<td>88.929</td>
<td>88.929</td>
<td>88.929</td>
<td>88.929</td>
<td>88.929</td>
</tr>
<tr>
<td>1100</td>
<td>4.400</td>
<td>91.316</td>
<td>91.316</td>
<td>91.316</td>
<td>91.316</td>
<td>91.316</td>
<td>91.316</td>
</tr>
<tr>
<td>1110</td>
<td>4.440</td>
<td>93.269</td>
<td>93.269</td>
<td>93.269</td>
<td>93.269</td>
<td>93.269</td>
<td>93.269</td>
</tr>
<tr>
<td>1120</td>
<td>4.480</td>
<td>95.439</td>
<td>95.439</td>
<td>95.439</td>
<td>95.439</td>
<td>95.439</td>
<td>95.439</td>
</tr>
<tr>
<td>1130</td>
<td>4.520</td>
<td>97.609</td>
<td>97.609</td>
<td>97.609</td>
<td>97.609</td>
<td>97.609</td>
<td>97.609</td>
</tr>
<tr>
<td>1150</td>
<td>4.600</td>
<td>infeasible</td>
<td>infeasible</td>
<td>infeasible</td>
<td>infeasible</td>
<td>infeasible</td>
<td>infeasible</td>
</tr>
</tbody>
</table>
A SUGGESTED NETWORK
B SYMMETRIC NETWORK
C FREE NETWORK

FIGURE 3.4.2B
TOTAL COST AS A FUNCTION OF TOTAL LOAD FOR THREE NETWORK CONFIGURATIONS
analogous critical flow for the network sets in at a level of about 600 vehicles input at each entry. The "softening" effect arises from a limited ability of the flow to use more than one outgoing lane from the entry node, and so to utilize several cheap arcs before using a costly arc.

Note, however, that in Table 3.4.2 (1), the critical flow level for lanes, \( f_c = 500 \), is the point at which the various solutions begin to diverge. It is therefore clear that the nonlinearity of the lane cost function (reflected in the limited capacity of the cheap arc) exerts a strong influence on the network solution.

We can rate network solutions according to two figures of merit:

(a) the total flow transmittable, i.e., the flow infeasibility point,

(b) the ratio of total cost to total load.

The "Suggested network solution", curve A, turns out to be the worst by both standards. Note that it begins to show the effects of congestion at precisely the lane critical flow point, earlier than the other solutions.

The "free network" is the best solution. Allowing capacity to be reallocated between every lane pair raises the feasibility limit from 3800 vehicles to 4560 vehicles, a gain of 19%. The total cost function is lowered at all points beyond the "congestion point" - a point which may be defined for the network as the point at which curves B and C diverge. At the maximum loading for the symmetric network, 3800 vehicles, the cost is 73,535 minutes. The free network cost at this point is 62,239 minutes, a gain of about 18%. The relative gain of about the same magnitude on both figures of
merit for the free solution, is an interesting similarity. We have not had time to check whether the following conjecture is generally valid, namely: the relative gain in network capacity, and the relative gain in system cost, from the optimization of capacity allocation, will be of the same order. More information can be gained about the properties of the network flow solutions by considering Figures 3.4.2 C and 3.4.2 D.

Figure 3.4.2 C shows the optimal flow pattern for the symmetric network. The heavy arrows show the lanes for which the flows exceed the capacity of the cheap arcs. Note that there is a flow in both directions between almost every node pair, indicating a fairly uniform utilization of capacity in an optimal solution. The exceptions occur on the lanes between nodes 7 and 8 (we can use the notation (7,8) for this lane) and (9,8) where in each case there is a flow from the former node to the latter, and no flow in the opposite direction.

Compare this solution with the similar diagram Figure 3.4.2 D for the free network. In this diagram the heavy arrows represent the directions in which capacity has been allocated to permit a flow exceeding the capacity of one lane. Note particularly that the heavy arrows in the free network correspond to those in the directed network, with the exceptions of lanes (9,6), (7,8), and (9,8). For these lanes, the flow does not exceed the single lane capacity of 760 veh/hr, but still exceeds in each case the "cheap arc" capacity.

The conclusion is that allowing the capacities to vary within a bundle will result in a capacity reallocation in which the lanes where costly arcs were utilized are favored
Figure 3.4.2C ISOTROPIC LOAD, LEVEL 950, FLOWS ON SYMMETRIC NETWORK
$n:f(k)$  FLOW ON ARC $n$ FOR COMMODITY $k$

$f(k)$  FLOW FOR COMMODITY $k$

FLOW $\leq$ SINGLE LANE CAPACITY

FLOW $<$ SINGLE LANE CAPACITY BUT

FLOW $>$ SINGLE LANE CAPACITY OF 760 VEH/HR

FLOW $\geq$ ODD ARC CAPACITY

FIGURE 3.4.2D  ISOTROPIC LOAD, LEVEL 950, FLOWS ON FREE NETWORK
with extra capacity. One might be led to conclude that the algorithm attempts to eliminate flows on costly arcs, and hence avoid the nonlinearity of the cost function.

But a further examination of Figure 3.4.2 D shows that, on the contrary, additional capacity is allocated to certain lanes, and then these lanes are heavily utilized.

The algorithm therefore constructs efficient "new" routes and capitalizes on this construction by reassigning flows. The algorithm does not leave old flow pattern essentially fixed, and merely remove the nonlinear cost components.

Note further that in the free network the set of roads used in only one direction is different from that in the symmetric network. Lanes (7,8) and (9,8) do not have zero opposite flow in the free solution, on the other hand, lane (2,5) now does.

Finally, in Figure 3.4.2 D the circled arrows represent flows which are not greater than single lane capacity, yet they are almost double the flows in the opposite lanes. We class the black arrows and the circled white arrows together as representing dominant flow directions in that solution.

This interpretation of dominant flow directions coincides with the diagram of Figure 3.4.2 E, where the directions which received the greater share of bundle capacity were marked with black arrows. That is, those flow directions received the greater share of capacity, for which the optimal flow either exceeded single lane capacity (this is an obvious necessity) or clearly exceeded the flow in the opposite direction.

Note in Figure 3.4.2 E that except for the single road (2,5) which had flow in only one direction, the relative al-
FIGURE 3.4.2E  ISOTROPIC LOAD, FREE NETWORK ORIENTATION AT LEVEL 950

VALUE OF DECISION VARIABLE FOR LANE, MINOR DIRECTION

VALUE OF DECISION VARIABLE FOR LANE, MAJOR DIRECTION
locations of capacities between two directions were clearly fractional e.g., 1.75:0.25, 1.5:0.5 were frequent ratios.

Summarizing to this point, we find that by passing from a symmetric network to one in which the capacities may be reassigned, the pattern of flows in the solution changes considerably. The system cost decreases, and the limit of feasible flows increases. Streets that had only one way traffic in the symmetric solution have two way traffic in the free solution, and vice versa.

We now move on to the question of finding the optimal one-way street pattern. As a first approach, we might use the "dominant direction" pattern of Figure 2.4.2 E as a hint for the structure of the one-way pattern (i.e., the "switching" solution). Suppose we arrange the streets so that they are one-way where a dominant direction is indicated in these figures and two-way elsewhere.

By a remarkable coincidence this pattern is exactly the "suggested" network of Figure 3.4.2 A. We say coincidence because this network configuration was suggested solely on the basis of the loading pattern and with no information on resultant flows. However, we have seen that this was a very poor solution. This means that rounding the decision variables to the nearest integer is not fruitful procedure for this problem.

The next method consisted of applying the branch and bound algorithm to obtain the optimal set of integer capacity allocation decision variables. Figure 3.4.2 F shows the tree diagram corresponding to this solution. However, this problem possesses an unusual special property: it is possible to add integer requirements in such a way that the objective function is not increased from the free solution. A complete integer
FIGURE 34.2F  ISO TROPIC LOAD, INTEGER PROGRAM

i: j  VARIABLE i,
    SET AT VALUE j
solution is obtained in which the objective function has the same value as for the free solution. But this integer solution corresponds to keeping all streets two-way except setting a one-way street from node 7 to node 8 and from node 9 to node 8. But this corresponds to a switching pattern which is obvious from the symmetric solution. Lanes (8,7) and 8,9) were the only empty lanes in that solution. In this solution the only cost saving occurs from doubling the capacities of the cheap arcs of lanes (7,8) and (9,8), eliminating the congestion costs on those lanes.

One might be tempted to conclude that the solution technique is obvious: one simple sets up a symmetric network and eliminates the empty or nearly empty lanes. But how do we know if this is in fact an optimal solution? Only by comparing it with the actual integer programming solution can we be sure. In this case, the free solution represents the absolute minimal cost flow and orientation solution for the network.

If we had set (7,8) and (9,8) one-way and evaluated the configuration, we would have a flow cost equal to the minimum obtained in the free solution. Since we know no other cheaper solution can exist, we know that this one-way pattern (which, admittedly, might have been guessed from the symmetric solution) is optimum. But it was necessary to solve the free optimum problem to obtain the lower bound on possible objective values.

A priori, it is not at all obvious why the limited change in the configuration which yields the optimum should give the same objective value as obtained when one has freedom to vary all capacities (within the fixed bundle capacity constraints). We suspect that the isotropy and high level of the loading leaves very little room for variations in the
configuration. Setting up one-way street clearly introduces asymmetries into the capacity configuration, and feasibility conditions can easily be violated by an incorrect pattern. In other words, the feasibility requirements greatly restrict the domain of variations desired for the goal of cost improvements.

3.4.3 The biased loading pattern

The coincidence of equal cost functions for the free solution and the optimal one-way solution was fortuitous. The next class of problems to be explored we term "biased". The load of commodity 3 was reduced by half in order to introduce asymmetry into the loading pattern and permit more latitude in road orientation. This had the effect of producing a free solution whose objective was less than the objective obtainable in an integer (switching) solution. Therefore, if we optimize the flows in the suggested pattern which has only (7,8) and (9,8) one-way, and if we solve for the optimal free network, we would obtain different objective values for these two solutions. In this case it is impossible to know immediately whether the suggested pattern is the best obtainable, or whether there lies another pattern whose objective lies closer to the free solution objective. In this situation it is essential to solve integer program via the branch and bound algorithm to prove that there is no solution with a lower objective. The execution of the algorithm is diagrammed in Figure 3.4.3 A. Note that the decision variables are clearly non-integral in the initial, free, solution. After a number of integral requirements have been added, many decision variables assume values within 10% of integers. When these variables are required to assume the nearest integer values, a reasonably good integer solution results. This means that a good solution can
i:j VARIABLE i, SET AT VALUE j
A, B, C, D OPTIMA

FIGURE 3.4.3A BIASED LOAD, INTEGER PROGRAM
be guessed at midway through the algorithm. However, the algorithm must be **completed** to determine the true optimal solution. The algorithm yields as an optimum the same solution as we found for the isotropic loading. Completing the algorithm shows that there are four alternative optimum solutions, labelled A, B, C, and D. These optimal switching patterns are described by the vectors of switching variables:

\[
A = (2, 1, 0, 2, 1, 1, 2, 0, 1, 2, 0, 1)
\]

\[
B = (2, 1, 0, 2, 1, 2, 2, 0, 1, 2, 0, 1)
\]

\[
C = (2, 0, 1, 2, 1, 1, 2, 0, 1, 1, 1, 1)
\]

\[
D = (0, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1)
\]

where the value of i'th component is the number of capacity units (lanes) assigned to the lane i position.

These solutions are diagrammed in Figures 3.4.3 B, C, D, and E.

Our computational experience with the symmetric case suggests a heuristic technique for optimizing the orientations. The technique consists of arranging the orientations so as to minimize the disturbance to the flow pattern which is optimal on the symmetric network, thus avoiding extensive rerouting and lengthening of routes. This means that one seeks routes with very light (preferably zero) flows in the minor direction as candidates to be one-way streets in the major direction.

The above heuristic can be placed in opposition to one based on choosing the most heavily loaded links (in the major direction) as candidates to be one-way streets in the major direction. The latter heuristic resembles the technique of solving the optimal flow problem on the free network, and
BIASED LOAD, INTEGER SOLUTION A

FIGURE 3.4.3B
FIGURE 3.4.3C  BIASED LOAD, INTEGER SOLUTION B
FIGURE 3.4.3D BIASED LOAD, INTEGER SOLUTION C
FIGURE 3.4.3E  BIASED LOAD, INTEGER SOLUTION D
rounding decision variables to the nearest integer value. In our example this lead to the "suggested network", a poor solution.

We should point out that in general, in mathematical programming, there may be more than one solution yielding the same optimal objective value. In the case of the isotropic loading example, we terminated the integer programming algorithm after obtaining only one solution. This solution was known to be optimal because it had the same objective value as the free solution.

In the biased loading problem, on the other hand, we were not able to construct an integer solution which had the same objective as the free solution for the same problem. This required us to complete the tree diagram. This tree contains four configurations of one-way streets which permit the same optimal flow cost.

The relation of these solutions to the one known solution for the symmetric loading example, and to the "suggested solution" to that example, is quite interesting. First of all, we observe that one of the solutions to the biased problem, solution D, is identical to the solution obtained for the isotropic problem. A parametric check showed in fact that this solution was optimal (in the sense of equaling the free network solution objective) for the entire range of symmetric loadings.

The other solutions A, B, C, to the biased problem bear a clear resemblance to the "suggested solution" for the asymmetric loading, and the "rounded decision variable" solution for both the symmetric and the biased loading problems. The resemblance is based on the existence of loop
systems of various size on the network. However, the "suggested solution", consisting of two loops which contain, between them, all except two arcs, is not an optimal solution.

The conclusion is that loop patterns (under these loading conditions) tend to produce efficient flow patterns, but may tend to prohibit feasibility, if constructed on a scale large enough to cause extensive route diversion. These results suggest that limited patterns in large networks would be efficient, provided they do not cause extensive route diversion and do not interfere with flow feasibility. Since the loop solutions are closely related to the solution obtained by rounding the decision variables, an alternative heuristic is suggested: find the pattern suggested by rounding, and study the behavior of the objective as the number of one-way streets is reduced. In a large system this may still be a formidable combinatorial problem, and it is not clear whether this technique would be an improvement over optimizing the integer program.

The question also arises, whether any of the alternative solutions B, C, D, to the biased problem are also solutions to the symmetric problem. We checked solution A, and in fact, this is also a solution to the isotropic problem, over the entire loading range. We did not have the opportunity to check solutions B and C.

In another sense, it is significant that reducing one of the commodity loads by a factor of one half to obtain the biased solution does not produce a significant change in the solution basis. This reduction is not sufficient to reveal a dominant set of routes which clearly determine the orienta-
FLOW 
SINGLE LANE CAPACITY OF 760 VEH/HR

FIGURE 3.4.3F  
BIASED LOADS, FREE OPTIMAL FLOW PATTERN
FIGURE 3.4.3G  BIASED LOAD, FREE NETWORK, OPTIMAL DECISION VARIABLES
FIGURE 3.4.3H  
BIASED LOAD, ORIENTATION RESULTING FROM ROUNDED FREE OPTIMAL DECISION VARIABLES
tations for the solution. For example, the total loading of commodity one, the "radial" flow, is at least double the load for each other commodity, yet we do not find a clear-cut lining up of capacity in favor of the radial direction.

3.4.4 The lopsided loading pattern

In order to produce a more clear-cut pattern in the flow requirements, we reduced the input of commodity 3 to zero. We call this the "lopsided" loading: there are only the radial loading and one transverse loading.

Figure 3.4.4A shows the optimal flow solution for the symmetric network. The large number of lanes with no flow in this solution suggests that it will be very easy to construct a one-way configuration. This was in fact the case. Figure 3.4.4B shows the free network solution; again there are many lanes with zero flow. There is a close but not exact correspondence between the null-flows in the symmetric and in the free solutions.

Figures 3.4.4C and 3.4.4D show the optimal free configuration and the rounded solution. There is again a close but not exact correspondence. An optimal integer configuration and the related flows are shown in Figures 3.4.4E and 3.4.4F. There are many arcs in corresponding directions and several which are not.

We arrived at an optimal solution very quickly, in the following way. Following the branch and bound algorithm, we solved the free network problem obtaining a total cost of 30399. In this solution decision variable 9 took the value 0.3. We chose to branch about this point. We solved the network problem with \( d_9 = 0 \) and \( d_9 = 1 \). With \( d_9 = 0 \), the
FIGURE 3.4.4 A  
LOPSIDED LOADING, FLOWS ON SYMMETRIC NETWORK
FIGURE 3.4.4B  LOPSIDED LOADING, FLOWS ON FREE NETWORK
ARC WITH NULL FLOW AND POSITIVE CAPACITY

FIGURE 3.4.4C  LOPSIDED LOADING, OPTIMAL FREE CONFIGURATION
FIGURE 3.4.4D  LOPSIDED LOADING, OPTIMAL FREE CONFIGURATION ROUNDED.
FIGURE 3.4.4E  LOPSIDED LOADING, OPTIMAL INTEGER CONFIGURATION
FLOW > SINGLE LANE CAPACITY OF 760 VEH/HR

FIGURE 3.4.4F  LOPSIDED LOADING, FLOWS ON OPTIMAL INTEGER CONFIGURATION
total cost is again 30399, and in this solution all decision variables have swung close to integer values. We round them all to nearest integer values; and the next solution is a complete integer solution with the same objective function. Having obtained one integer solution we terminated the algorithm. A completed algorithm would undoubtedly have revealed many alternative solutions.

We conclude from this computation that where the loading pattern shows a strong asymmetry the free network lies close to an optimal integer configuration, and the integer solution is quickly obtained. The clearer the directionality of the flow pattern, the easier it is to determine the optimal network configuration.

3.4.5 The one-way street problem: conclusions

We have found the branch and bound method of integer programming to be an effective technique for determining optimal configurations of one-way streets. The results obtained agree with intuition, but these results cannot be guessed in advance. An applicable heuristic approach is to solve the flow problem on the symmetric network and on the free network. Next study the symmetric network for lanes with very small flows. These are the most likely candidates for one-way streets in an optimal (or near optimal) solution. This "suggested" solution can be evaluated by solving a linear program with integer requirements.

If the resultant objective function is equal to the objective function for the free solution, we have a known optimal configuration. If it is near the free objective function, we have a "good" configuration.
We have seen that flow feasibility requirements are extremely important in restricting the types of configurations which yield good solutions. Loop configurations were suggested as good solutions by the flow patterns, but unless restricted in their extent, they caused inefficient route diversions.

3.5 The Link Addition Problem

We now proceed to another application of the "integer programming on networks" technique.

In this application we are given a present network configuration, with known loads and travel costs. Travel costs on some lanes are represented by arc pairs, and on other lanes by single arcs, according as the lanes are likely to suffer congestion (hence non-linear costs) or not. Various lanes are proposed as feasible additions to the network, at known construction costs, pro-rated to the flow period. There is a constraint on the total construction cost. The problem is to determine the configuration (i.e., the construction plan) which minimizes the sum of flow (i.e., road users') and construction costs.

Professor P. O. Roberts, Jr., (4) was studying a problem of this form independently of the present writer. He found that the Gomory algorithm for integer programming (5) in its SHARE distributed version (6) would not solve the problem. That is, the computations did not converge in the computation time available. Applying the branch and bound algorithm, we easily solved a problem in the form given by Roberts.

3.5.1 An example and its solution

Figure 3.5.1 A illustrates the example problem.
We have a seven node network. The existing lanes are as shown. To simplify the problem, by reducing the number of lanes, there is flow permitted in only one direction between most node pairs. The input nodes are nodes 2, 5, and 7, at the right, so that the dominant flow direction is to the left. Links A, B, C, and D are proposed. Which construction plan affords the minimum total cost? Figure 3.5.1 B shows the resolution of the problem by the branch and bound technique. This example was chosen by Roberts to be easily calculable without the benefit of integer programming. These are only sixteen possible integer solutions; some of these are ruled out by limitations on the construction budget. By solving the linear programs corresponding to the permissible decisions, the optimal plan can be calculated in less time than it takes to execute the branch and bound algorithm. However, when the number of decision variables in large, direct enumeration of solutions becomes impossible, as we pointed out in the one-way street problem. The three best solutions are: construct A (objective 3365), construct C and D, (objective 3389), construct B (objective 3395). Note that C and D together form a good solution, but the tree diagram shows that building C and not D yields an objective worse than 3403 (given the best choice on A). This simple example shows how a good solution may be overlooked if only the construction of single links (i.e., taken one at a time) is evaluated. This emphasizes the need for a study of over-all system effects of decisions - the study of combinations of alternatives and their over-all consequences - rather than the determination of local priorities.

We remark, finally, that the order of solutions, A, C, D, B, is not at all obvious from the loading pattern. The heaviest inputs appear at nodes 5 and 7. One might think B
\( m(t) \) | \( m \) UNITS COMMODITY \( i \) INPUT/OUTPUT
\((j,k) \leq c\) | ARC \( j \), USER COST \( k \), CAPACITY LIMIT \( c \)
\( p \) | CONSTRUCTION PRICE

\* | COMMODITY 1, 2 ONLY
\* \* | COMMODITY 3 ONLY
----- | PROJECTED LANE

FIGURE 3.5.1A  LINK ADDITION EXAMPLE
**FIGURE 3.5.1B** LINK ADDITION EXAMPLE, INTEGER PROGRAM
would have the highest priority, or possibly CD. It is clear from the diagram, in this simple case, that an improvement at D is useless without an improvement at C, since the pair of arcs is necessary to permit 6 to be used as a transhipment node. From the loading diagram, link A appears to be of low priority. However, its low construction cost compensates for its mediocre location - an effect difficult to determine without solving the mathematical program.

3.6 Summary

We have formulated three types of traffic network synthesis problems as mixed integer programs, and solved examples of two of these three. The branch and bound technique was used to determine an optimal solution (in certain cases, all optimal solutions) to families of these problems. The algorithm permits the rapid construction of a good solution; the proof that this is an optimal solution may take considerably more effort. The time-sharing computer facility was invaluable in permitting one to evaluate and control the progress of the algorithm. The character of the solutions suggested an a priori heuristic approach: beware above all of violating feasibility requirements. In seeking a good network configuration, the safe approach is to minimize the penalties attached to any decision, rather than boldly striking out to maximizing the gain. This strategy is analogous to the optimal strategy in a zero-sum two person game, where each player should strive to minimize his loss at each play. This strategy is rigorously optimal for such a game, but we propose the network analogue simply as a heuristic approach.
3.7 Notes to Chapter III


CHAPTER IV
Strategies, Decomposition, and Priorities in the Multi-
Commodity Flow Problem

4.1 Introduction

In Chapter I we referred to the work of Charnes-
Cooper (1) in which traffic flow on a network was charac-
terized as a game among players associated with the ori-
gins. Each player determines a set of routes, that is, a routing plan or strategy, given certain information about the travel costs in a network. Each player attempts to minimize the total travel time to destinations for the vehicles departing from his origin.

Beckman, McGuire, and Winsten (2) discussed the "problem of Pigou" for a pair of alternate links between one origin and one destination node. Their argument was that when each vehicle seeks to minimize its own travel time, regardless of the congestion on a link, social dis-
utilities are incurred because a vehicle added to a con-
gested link causes losses to all other vehicles on that link. That additional vehicle should be forced to select the non-congested link, even at a slightly greater travel cost to itself, for the benefit of the total system. This rerouting was to be encouraged by the levying of tolls on links congested due to high demand. These tolls would en-
able the social system to recapture from an individual ve-
hicle the disutility caused to the system (i.e., all other vehicles on the congested route) by its entry on the route.

Beckman et al were therefore interested in modifying the independent route choices made by vehicles seeking
shortest routes, in order that a system optimum be obtained. In the game proposed by Charnes-Cooper, we also have a process in which route selections are made so as to optimize costs from the viewpoint of the several individual players. We pass from Beckman's formulation to that of Charnes simply by associating the route decision not with each vehicle, but with the class of all vehicles leaving a particular origin.

What form does the problem of Pigou take in the game theoretical model? Can we show how independent decisions produce flow patterns which are not system optimal? Can we show how the proper taxation forces decisions to produce a system optimal flow pattern?

In this chapter we shall illustrate the problem of Pigou on a network which differs slightly from that used in Beckman's example. In that example, there were two parallel links. One had low user cost and low capacity, the other had higher user cost and high capacity. The problem was to divert some traffic from the link of low user cost, to avoid congestion.

Our example network will be described below. The problem will be to study the flows on this network in terms of the game theoretic model of Charnes-Cooper, and to show how independent decisions do produce non-optimal flow patterns. Very simple examples will illustrate the stability (and instability) characteristics of solutions to this game.

To pass from the independent decisions in this game to a solution in terms of directed decisions which produces a system optimum solution, we apply the decomposition algorithm of linear programming. There is a natural transition from the statement of the game problem in terms of alter-
native strategies to a statement in terms of decomposition theory. For a general discussion of the Decomposition algorithm, the reader is referred to Dantzig (3). Our treatment will cite only the main applicable results. C. Pinnell (4) applied the decomposition algorithm to a multicommodity network flow problem. Pinnell was entirely concerned with the development of an efficient computer program for producing a system optimum flow pattern. Pinnell did not analyze the decomposition algorithm for its implications regarding strategies (Charnes) and tolls (Beckman). After much computation, Pinnell observed that the order in which players were allowed to determine their strategies (Pinnell used a different terminology) affected the rate at which the computer program converged to a system optimum solution. Pinnell therefore observed the role of priority in the route allocation process.

We intend to apply the decomposition algorithm to the problem of optimal strategies on our sample network. Our interpretation of this simple example will unify the game model (Charnes) the effect of tolls (Beckman), and will permit us to develop naturally a general interpretation of priority in multicommodity flows. This interpretation will lead us to a heuristic technique for controlling traffic on the network. Our analysis of tolls in the decomposition algorithm is an application to the network flow problem of the ideas developed for the general linear programming problem by Fabian and Baumol (5).

There are four other background works for our analysis. In chronological order, they are Koopmans (6), Jewell (7), Jorgensen (8), and Dantzig (9).
Koopmans showed that a basic solution to the linear program of the transportation type has the form of flows along a tree in the transportation network. A tree is a subset of arcs of the network such that there is exactly one path between any two nodes (i.e., there are no loops and the graph is connected).

Dantzig illustrated this property for a single commodity transhipment problem. If we have a transhipment problem without capacity constraints on the arcs, then each basic feasible solution corresponds to a tree flow. The optimal solution will consist of the tree made up of the single shortest route from each origin to each destination. If there is only one destination, then the solution is a minimum length tree routed at the destination. If there are capacity constraints in the transhipment problem, then the solution will still be a tree, but not necessarily the minimum path tree. There is competition among the sources for routes of limited capacity and the solution is a total system optimum, so that the flow from certain origins may be forced to take longer routes.

In our example problem we will have two commodities. The flow problem for each commodity viewed separately is a transhipment problem. We are free to interpret the capacity constraints as follows: capacity bounds are shared by the two commodities, and there are no explicit capacity bounds on the flow of a single commodity.* Under these conditions, each basic feasible solution to the single commodity problem will be a tree on the network, and the optimal solution will be the tree, rooted at the destination, of shortest routes to sources. When taken together, the two optimal solutions to the single commodity subproblems may violate

* The flow problem for each commodity can thus be stated as if it possessed no capacity limits; the capacity limits on flow of all commodities in each arc will ultimately restrict the solutions to the single commodity problems.
the common capacity constraints and hence may not be a feasible solution to the two commodity problem.

The decomposition approach which we shall apply constructs feasible solutions to the two commodity program out of basic feasible solutions (i.e., trees) of the single commodity problems.

Jewell, on the other hand, studied the multi-commodity flow problem using a primal-dual technique. He treated each single commodity problem as a bounded transhipment problem, and obtained a sequence of primal infeasible but dual feasible solutions. He did this by building up flow from zero for each network copy independently, maintaining dual feasibility for that copy. When the individual copy loads became great enough so that common capacity constraints become binding, the common capacities may be shifted to one commodity or another, to permit convergence to primal feasibility.

The rules for accomplishing this shifting are expressed in a rather involved extension of the primal-dual algorithm for the transportation problem.

However, Jewell's work interests us mainly because his approach to the multi-commodity flow problem emphasizes the competition for routes of limited capacity among commodities as an essential component of the optimal route allocation process.

The decomposition algorithm permits us to illustrate this competition for routes in terms of primal feasible solutions to the single commodity non-capacited minimal cost flow problems, that is, in terms of minimum path trees.
We shall show how the optimum solution is not that in which each commodity follows its own shortest route strategy, when capacity constraints are binding, and travel cost is an increasing function of flow on arc. We approximate the non-linear travel cost on a lane by our old trick of simulating the lane by two or more parallel arcs which have increasing costs. By "discretizing" the non-linearities of the cost function, we shall obtain a clear representation of the relation between the competition for capacity, the assignment of shortest routes to certain commodities according to an optimum priority schedule, and the denial of shortest route facilities to other commodities.

The relation between minimum path solutions and minimum total cost solutions was studied by Jorgensen for networks in which travel time was a continuously differentiable convex function of flow. Using the Kuhn-Tucker conditions for optimality in a non-linear program, Jorgensen showed how the system optimum differs from the individual path solution when costs are not constant.

Our argument will yield an equivalent result, but there are two advantages in using the "discrete travel time function" - i.e., parallel arcs. The first is that the priority process becomes very clear, and the second is that the decomposition algorithm for linear programs becomes available as a computational procedure.

Having sketched the range of topics to be covered in this chapter and reviewed the sources of our ideas, we proceed to describe the example problem which will serve as a basis for all the analyses in this chapter.
The motivation for these analyses can be summarized briefly as follows. An assignment by the minimum path technique consists of the sequential loading of transfer volumes on the network along paths which are of minimal time cost at that stage in the calculational sequence. In an assignment program patterned after the Charnes-Cooper model, the pattern of loading are periodically (iteratively) revised according to information on the time costs on the network. Revision of a path involves selection of new minimum path routes. In the decomposition formulation of the assignment problem the flow pattern is also revised by the selection of new routes. In this formulation the routes are chosen not in terms of the "actual" time costs, but in terms of time costs biased by the simplex multipliers, or "tolls". The decomposition formulation is a restatement of the linear program for finding a system optimal flow pattern; in this formulation the calculation proceeds in a manner parallel to the minimum path assignment technique. Comparing the minimum path assignment technique and the system optimum technique in decomposition form sheds new light on their similarities and differences.
4.2 The Routing Plan Example

Figure 4.2 A shows the example network.

There are two commodities flowing. The first commodity has an input of one unit at node 1 and an output of one unit at node 3. The second commodity has an input of one unit also at node 1, and an output of one unit at node 2. We associate player, or strategist, M, with commodity 1, and player N with commodity 2. The players are associated with destinations rather than origins as in the Charnes model, but this makes no difference in the analysis. We can therefore visualize player M sitting at node 3, who must choose some route for shipping one unit of flow from node 1 to node 3. Player N at node 2 must choose route for shipping a unit of flow from node 1 to node 2.

We wish to devise a set of rules (a game) which defines the manner in which M and N make their decisions.

Before we do this, let us analyze this simple network. Arc 1 (cost 1) and arc 2 (cost 2) can be considered as forming an arc pair which represents a lane which has a non-linear cost function. The optimal plan from the viewpoint of player M consists of flowing one unit along arc 1 and then along arc 4. This is the shortest route from node 1 to node 3. This saturates arcs 1 and 4. The shipping cost to M for this plan is 2. The optimal plan for player N consists of shipping one unit over arc 1 at a cost of 1; this is the shortest route from node 1 to node 2. However, if player M is allowed first route choice, he occupies arc 1, and the optimal plan for player N is denied him. His second choice is arc 2 at a cost of 2. Arc 2 is the shortest available route. The costs under this arrangement are 2 to M and 2 to N for a system cost of 4. If on the other hand, player N had
FIGURE 4.2A  EXAMPLE NETWORK

INPUT/OUTPUT I UNIT, COMMODITY K
(i,j)  ARC i, USER COST j
CAPACITY = 1, ALL ARCS
first choice, he would occupy arc 1 at a cost of 1. Player M would then chose arc 3 as the shortest available route; the cost of this plan is 2.5. Under these strategies, the system cost is 3.5, less than the previously determined system cost. It is obvious under these circumstances that each player's seeking to minimize his own route cost results in a non optimal solution for the system as a whole. The crucial arc is arc 1: priority in choosing a route containing arc 1 directly affects the type of solution we obtain.

We stated above that the basic feasible solutions to the individual flow problems can be represented as rooted trees on the network. For a network as simple as this, we can easily enumerate the trees for each commodity. Figure 4.2 B shows the basic trees for commodity M, each representing a 'basic' shipping plan for player M. Figure 4.2 C shows the corresponding information for player N. The cost $P_{Mi}$ for a plan $T_{Mi}$ is calculated by summing the product of the flow on each arc times the unit user cost on that arc.

These basic solutions to the single commodity transhipment problem are degenerate - that is, some basic variables will appear at zero level in the solution. Graphically this means that a feasible flow pattern can be found with no circulatory loops, but the flow pattern is not a tree because the flows in some arcs properly belonging to the tree are at zero value. In order to remove the degeneracy, we construct a complete tree by adding an infinitesimal flow $e$ along arcs chosen because they connect the graph and form no loops. The amount $e$ is input to and output from the network at the proper points to preserve conservation of flow conditions, and $e$ figures in the cost $P_{Mi}$. 
PLAN $T^N_1$
COST $P^N_1 = 1 + 2.5e$

PLAN $T^N_2$
COST $P^N_2 = 2 + 2.5e$

PLAN $T^N_3$
COST $P^N_3 = 3.5$

FIGURE 4.2B  BASIC SOLUTIONS FOR COMMODITY $N$
FIGURE 4.2C  BASIC SOLUTIONS FOR COMMODITY M

PLAN T^{M_1}
COST P^{M_1} = 2.5 + \varepsilon

PLAN T^{M_2}
COST P^{M_2} = 2

PLAN T^{M_3}
COST P^{M_3} = 3
Introduction of the flows $e$ amounts to a perturbation of the right hand side vector of the linear program (perturbing the input output vector). This is a standard technique for removing degeneracy in linear programming. It is useful for preventing cycling in the simplex algorithm, for proving convergence, and, in our case, for demonstrating that each basic solution corresponds to an actual tree. However, in our simple example, this perturbation is inessential, and we can set $e = 0$.

Each routing plan can be described by a column vector which has one component for each arc in the network. The value of the $i$-th component is the level of flow on the $i$'th arc in the corresponding shipping plan. Therefore the plans in Figure 4.2 B and 4.2 C can be described by the following vectors (* denotes transpose):

<table>
<thead>
<tr>
<th>Plans:</th>
<th>Costs:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{M1}^*$ = (e, 0, 1, 0, 0)</td>
<td>$P_{M1}^*$ = 2.5 + e</td>
</tr>
<tr>
<td>$T_{M2}^*$ = (1, 0, 0, 0, 5)</td>
<td>$P_{M2}^*$ = 2</td>
</tr>
<tr>
<td>$T_{M3}^*$ = (0, 1, 0, 0, 5)</td>
<td>$P_{M3}^*$ = 3</td>
</tr>
<tr>
<td>$T_{N1}^*$ = (1, 0, e, 0, 0)</td>
<td>$P_{N1}^*$ = 1 + 2.5 $e$</td>
</tr>
<tr>
<td>$T_{N2}^*$ = (0, 1, e, 0, 0)</td>
<td>$P_{N2}^*$ = 2 + 2.5 $e$</td>
</tr>
<tr>
<td>$T_{N3}^*$ = (0, 0, 1, 1, 0)</td>
<td>$P_{N3}^*$ = 3.5</td>
</tr>
</tbody>
</table>

where $e$ may be set equal to zero.

We can succinctly restate our previous discussion in terms of these vectors: If $M$ has priority, he chooses plan $T_{M2}^*$ (cost 2) and $N$ will choose $T_{N2}^*$ (cost 2). The system cost is 4. If $N$ has priority, he chooses $T_{N1}^*$ (cost 1), and $M$ chooses $T_{M1}^*$ (cost 2.5). The system cost is 3.5. Therefore by giving route priority to $N$, we achieve a system optimum flow pattern.
4.3 Game Theoretical Interpretations

Before we show how the decomposition algorithm and its pricing mechanism determine this optimum pattern, we shall illustrate two versions of the Charnes-Cooper multi-person game with our example.

Charnes-Cooper assumed that each commodity tends to flow along the shortest (time) path available. But the time length of a path depends on the loading of the link, so that the decision of a player as to his route choice (plan) depends on the decisions of other players. Charnes-Cooper proposed that by iterating the plan decisions, the problem would settle down into a stable flow pattern. As we pointed out in Chapter I, this idea lies at the bottom of most traffic assignment computer programs in current use.

As Charnes-Cooper originally proposed the game, the players in a fixed order would choose strategies given the state (loading) of the network as a result of the previous strategies. The new strategies determine a loading which replaces the old loading, and a new set of decisions are made. Charnes-Cooper posed the question: will the decision converge to a loading pattern stable against a single player's strategy change? against multi-player strategy change?

Note that the original formulation of the Charnes-Cooper game overlooks the problem of explicit capacity constraints.* In our model, we "discretize" the cost function, and the capacity constraints on arcs simulating a link are inextricably bound up with the notion of non-linear costs. The question arises, what provision do we make for capacity constraints in the Charnes-Cooper game?

* The link total travel costs were assumed asymptotic to an infinite cost at the capacity limit; flow would never reach the capacity limit. In the LP Model, the costs are finite up to the capacity bound; flows can reach but not exceed the bound because of the explicit linear constraint bounding flows. Because the explicit bound is computationally more convenient then an asymptotic cost, the former is used in the LP Model.
Early assignment programs replied to this question as follows: make the strategy decisions in order according to the previous network state, assign the flows according to these strategies, and then assign a large penalty cost to each strategy involved in the violation of a capacity constraint. Next go back and permit the players to revise their strategies.*

Using our simple example, we can easily see why such a game leads to unstable solutions. We can describe the game as the following sequence of "plays".

Play 1: M: $T^{M2}$, cost 2
N: $T^{N1}$, cost 1
Each player chooses his shortest route assuming no loading on the network.

Evaluation: Capacity constraint on arc 1, (the most desirable arc) is violated. Assign penalty cost $P$ to flow on arc 1.

Play 2: M: $T^{M1}$, cost 2.5
N: $T^{N2}$, cost 2
Both players seek shortest route avoiding arc 1.

Evaluation: Arc 1 is now empty, and both players seek plans of lower cost than the current plans.

Play 3: M: $T^{M2}$, cost 2
N: $T^{N1}$, cost 1
Evaluation: Both players converge once again on arc 1.

Therefore the solution is cycling: the strategies oscillate between convergence on arc 1 and avoidance of arc 1. The players thus alternate

* Each revision by player i is a decision made on the assumption that other players will hold their strategies fixed. These decisions are "single player perturbations" of the game state.
between plans which are individually advantageous but mutually catastrophic, and plans individually disadvantageous.

The above game falls in the category of non-cooperative games with imperfect information. Suppose we now modify the game to reduce the imperfection in the information available to the players. To do this, we immediately warn a player $N$ if he is about to choose a plan which will violate a capacity constraint under all previous loading, including loadings in the current play by players ahead of player $N$ in the assignment order. In the assignment computer program terminology, this corresponds to "updating the network loading after every assignment".

Let us recommence the previous game, subject to the modified rule.

**Play 1:**

- **M:** $T^M_2$, cost 2, forbidding $T^N_1$ by saturating arc 1
- **N:** $T^N_2$, cost 2

**Evaluation:** Given this loading, M maintains $T^M_2$ since no alternative plan has lower cost. N must maintain $T^N_2$ for the same reason.

**Conclusion:** This game is stable against single person variations. Improving the distribution of information has prevented the cycling situation. It is also clear that if N were given first choice in selecting his plan, we would have a system optimum, hence stable solution.

Maintaining the plan selection order $M$, $N$, we ask if this game is stable against two-person variation, at the state $(T^M_2, T^N_2)$. We have assumed the two players do not cooperate.
Each player knows the other's current strategy. Player N must assume player M will maintain his current, individually optimal plan, so that N gains nothing from changing his plan. M obviously gains nothing from changing plan. So this game is stable against two-person non cooperative variation. The difference between the two-person variation and the one-person variation in our interpretation, lies in the fact that in the latter case, each player assumes the other will not change plan, and in the former case, each player assumes the other player might change plan.

Suppose, that we had, not a non-cooperative game, but one in which the players have perfect information and collaborate. Suppose the game find itself in state \((T^M_2, T^N_2)\). If the players collaborated, and agreed on transforming the game to the system optimum state \((T^M_1, T^N_1)\), player M's cost would increase by 0.5 and player N's cost would decrease by 1.0. If the players agreed to "split the profits", of this action, player N would "pay" player M 0.75, and the cost of each player would have decreased by 0.25.

This interpretation of cooperative games appears appropriate to the theory of collusion between firms, but does not appear to describe realistically a multi-commodity flow situation, where travel cost net losses and gains cannot be exchanged among vehicles (unless the system represents, for example, large trucking fleets, and not commuters' automobiles).

The example of the cooperative game was introduced mainly for the sake of contrast with the non-cooperative game, which we hold to represent the commuter traffic model. In the non-cooperative game, the solution will most likely not converge to a system optimum solution - unless by change
the (randomly allocated) plan selection priorities happen to be allocated in the way which leads to the optimum.

We conclude that the non-cooperative game which forms the basis for most traffic assignment computer programs, lacks the mechanism for arriving at a system optimum flow pattern. Moreover, versions of the program which do not frequently disseminate information on capacity limitation may be subject to oscillatory solutions. Extending the model to permit cooperation between players not only appears unrealistic as a model for commuter traffic, but appears subject to great computational difficulty when extended to many players. How is a computer program to model a game in which players agree on strategies which might lead to a system optimum? Is one to compare vast numbers of alternative strategies? This appears exceedingly costly. Instead, we shall apply the decomposition algorithm of linear programming to develop a method for recommending strategy changes to the players, in such a way that a system optimum is obtained.

Before we go on to the decomposition interpretation, we should like to make a few concluding comments on the game theoretical interpretations. There are general theorems on the existence of solutions which are stable against single person and multi-person strategy perturbation. We, obviously, have not proven any general theoretical results on the N person game as a model of multi-commodity traffic flow. But we have presented some simple examples which demonstrate the possibilities of stable or oscillatory system optimum or non-optimum solutions. We presented these models because we believe they represent the essential characteristics of traffic models built into the assignment programs. The behavior of these assignment programs with respect to stability has
not been systematically studied (or the results have not been published), and these simple examples illustrate starkly the behavior which occurs in computer programs involving hundreds or thousands of nodes, and thousands of route decisions.

When presenting their game theoretical model of multi-commodity flows, Charnes and Cooper refer to an equilibrium solution which is stable against deviations by a single player as a Nash equilibrium.

Nash's Theorem on n-Person Games (10) states that in an n person non-cooperative game with a finite number of strategies, such an equilibrium always exists. In fact, several such equilibria may exist and they may have different system cost values. However, the equilibrium may consist not of pure strategies, but of mixed strategies. A pure strategy in the network example is a single plan (single tree) for each player. A mixed strategy is a convex combination of pure strategies adopted by a player. The Game Theory views the mixed strategy as a probability distribution of pure strategies, with the cost function the expected value of the costs for the pure strategies.

In our example of two non-cooperative players choosing plans for flows on a network, in a fixed order of choice, we showed that after player M chooses plan $T^M_2$, player N chooses $T^N_2$ (unless forbidden by capacity constraint). This leads to an unstable (non-equilibrium) solution. If M chooses plan $T^M_1$, then N will choose plan $T^N_1$. We showed that this is a Nash equilibrium solution, whose existence is guaranteed by Nash's Theorem. In this particular example the equilibrium solution consists of a pure strategy for each player.
In terms of linear programming, we can view a mixed strategy as a deterministic, convex combination of pure strategies. For example, a player would send 30 % of his vehicles via tree $T^1$, and 70 % via tree $T^2$. A basic theorem of linear programming states that every point of a bounded convex region is expressable as a convex combination of extreme points. The extreme points correspond to basic solution, which for the transhipment problem are trees on the network. Therefore, in terms of the network flow problem, one has the theorem: every feasible flow is expressable as a convex combination (or weighted average) of tree flows.

Exploitation of this theorem forms the core of the decomposition algorithm. We shall pass from the game theoretical notion of mixed strategies, (there is at least one mixed strategy for which a Nash equilibrium exists) to the linear programming notion of convex combinations of extreme points, and we shall use this notion to obtain, not a Nash equilibrium solution to the traffic assignment problem, but a system optimum solution. This is the solution which would occur in a game theoretical model with perfect cooperation and perfect transferability of utility, as we suggested in one of our simple examples. We shall obtain the solution not by game theoretical means, but by linear programming. The linear program thus takes over the function of the cooperating and trading players, and assumes the role of an impartial arbiter of the game. The arbiter is to determine a set of tolls for the arcs in the network, which has the user costs as perceived by the players, in such a way that acting independently, they choose (pure or mixed) strategies which converge to a system optimum.
4.4 Decomposition and the Multi-Commodity Flow Problem

The decomposition algorithm is based on an extension and exploitation of the following two ideas, both of which were known before the decomposition algorithm was developed.

The first idea is that any interior point of a convex polyhedron can be expressed as a unique convex combination of extreme points. This means that any feasible solution of any linear program (with bounded solutions) can be expressed as a convex combination of basic solutions. The second idea is that in a revised simplex algorithm one only needs to have on hand one column of the constraint matrix at a time when searching for a column to enter into the basis.

These two ideas are exploited in the decomposition algorithm to gain both a reduction in the number of constraint equations, and a way to avoid writing out the entire constraint matrix in advance.

These effects can provide an immense gain in the size of problem which is computable, but we are going to apply the technique to our small example of previous section. The algorithm provides a demonstration of the development of a system optimal set of shipping plans.

In our example, we had two commodities, M, and N.

We can write the minimal cost flow problem as an "ordinary" linear program in terms of the flow vectors $F^M$ and $F^N$ (flow of commodity on each arc), the incidence matrix $M$ for the network, the flow input-output requirement vectors $R^M$ and $R^N$ (input/output of commodity at each node), the arc travel time vector $T$, and the arc capacity vector $C$. 
The linear program is

\[
\begin{bmatrix}
M & M \\
I & I \\
T & T
\end{bmatrix}
= \begin{bmatrix}
R^M \\
R^N \\
C
\end{bmatrix}
\]

(min)

In our example we have

\[
R^{M*} = (1, 0, -1)
\]

\[
R^{N*} = (1, -1, 0)
\]

\[
C^* = (1, 1, 1, 1, 1)
\]

\[
T^* = (1, 2, 2, 5, 1, 1)
\]

The decomposition algorithm begins with the observation that any solution \((F^M, F^N)\) to this "master" linear program must independently satisfy the separate "subprograms"

\[
MF^M = R^M
\]

and \(MF^N = R^N\)

in addition to simultaneously satisfying the common (or "corporate" constraints

\[
IF^M + IF^N \leq C
\]

or \(F^M + F^N \leq C\)
Now we make use of the first idea mentioned above, and express the fact that all feasible solutions to each subprogram can be expressed as a convex combination of the extreme points of that subprogram:

\[
F^M = \sum_{i} m_i T^{M_i}
\]

\[
F^N = \sum_{j} n_j T^{N_j}
\]

The extreme points to the subprograms are the basic solutions to transhipment problems $T^{M_i}$ and $T^{N_j}$. We have assumed we know all these at this time. This is true for our simple example, but for a large problem the enumeration of all trees is an enormous task. We shall see that it need not in fact be carried out.

The $m_i$ and the $n_j$ are weights, or percentages, which are attached to the various basic solutions $T^{M_i}$ and $T^{N_j}$ to the subprograms.

The next step involves substituting these expressions for $F^M$ and $F^N$ in the corporate constraint.

\[
F^M + F^N \leq C
\]

becomes

\[
\sum_{i} m_i T^{M_i} + \sum_{j} n_j T^{N_j} \leq C
\]
The $T^M_i$ and $T^N_j$ are fixed, "known" vectors. Therefore this inequality can be interpreted as a set of capacity constraints, one per arc, on new variables, the $m^i$ and $n^j$.

This suggests a new linear program in the variables $m^i$ and $n^j$; the columns of the constraint inequalities are formed by the basic solutions to the subprograms:

$$
\begin{align*}
T^M_1 & \quad T^M_2 \quad \ldots \quad T^M_k & \quad T^N_1 & \quad T^N_2 & \quad \ldots & \quad T^N_k \\
& \quad 1 & \quad 2 & \quad k & \quad 1 & \quad 2 & \quad k \\
M^i & \quad M^i & \quad \ldots & \quad M^i & \quad N^i & \quad N^i & \quad \ldots & \quad N^i
\end{align*}
$$

The objective row is made up of the cost attached to each subprogram basic solution. The cost $P^M_i$ is the sum of the arc flows times the arc costs, for the tree $T^M_i$. The row vector of arc costs $T$ has components $t_k$ and we can write

$$
P^M_i = T T^M_i = \sum_{k} t_k T^M_i
$$

where $T^M_i$ is the flow on the $k'$th arc are due to the $i'$th basic solution to the subprogram for commodity $M$. In our example we calculated tree costs in exactly this way, and obtained:

$$
P^{M1} = 2.5 \quad P^{M2} = 2 \quad P^{M3} = 3
$$

$$
P^{N1} = 1 \quad P^{N2} = 2 \quad P^{N3} = 3.5$$
For this transformed linear program in \( m^i \) and \( n^j \) to be equivalent to our original linear program in \( F^M \) and \( F^N \), we must add the "convexity constraints"

\[
\begin{align*}
& S) m^i = 1 \\
& i \\
& S) n^j = 1 \\
& j
\end{align*}
\]

to the transformed linear program. This assures that every solution \((M, N)\) to the transformed program determines pair of convex combination of basic solutions to the subprogram, and hence represents a feasible solution to the subprograms as well as the transformed master program.

With convexity constraints added, the transformed master program has the form

\[
\begin{pmatrix}
1 & m & 2 & \ldots & m & k & 1 & 2 & \ldots & n & k \\
\end{pmatrix}
\]

\[
T^M1 \quad T^M2 \quad \ldots \quad T^Mk \quad T^N1 \quad T^N2 \quad \ldots \quad T^Nk
\]

\[
\equiv C
\]

\[
\begin{pmatrix}
1 & 1 & \ldots & 1 & 0 & 0 & \ldots & 0 \\
0 & 0 & \ldots & 0 & 1 & 1 & \ldots & 1 \\
p^{M1} & p^{M2} & \ldots & p^{Mk} & p^{N1} & p^{N2} & \ldots & p^{Nk}
\end{pmatrix} = (\text{min})
\]

We can give the name \( M^i \) to the tree vector \( T^M^i \), augmented by a unit element in the convexity constraint row corresponding to commodity \( M \). Similarly we name an augmented \( T^N^i \) vector, \( N^i \). In other words, we augment \( T^M^i \) by a unit vector \( e^i \) in the space of convexity constraints whose dimensionality equals the number of subprograms.
\[ M^\star = (T^{Mi}, e^i)^\star \]
\[ N^\star = (T^{Nj}, e^j)^\star \]

Therefore, the decomposition algorithm transforms our original minimal cost flow program into the following linear program.

\[
\begin{align*}
& \sum_{i=1}^{m} m_i \cdot e_i \quad \sum_{j=1}^{k} n_j \cdot e_j \\
& = s_1 \quad s_2 \ldots \quad s_a \\
& \sum_{i=1}^{m} M^i \cdot e_i + \sum_{j=1}^{k} N^j \cdot e_j = I \\
& p_{M1} \cdot p_1 + p_{M2} \cdot p_2 + \ldots + p_{Mk} \cdot p_k = H
\end{align*}
\]

where we have transformed the inequalities into equalities by addition of the capacity slack vectors \( S_i \) for each of the \( a \) arcs, and \( C^1 \) is the vector \( C \) augmented by a unit component for each commodity.

In our original problem, the number of constraints was given by the product of the number of nodes (rows in the incidence matrix) and the number of commodities (number of blocks in the block diagonal matrix) plus the number of capacity constraints. In the transformed problem, we have one constraint for each capacity constraint, plus one convexity constraint for each commodity, instead of the number of rows in the incidence matrix for each commodity. It would appear that the price we pay for this is to introduce all basic solutions (all trees) for each commodity, instead of a column for each arc. The effect required to enumerate all these trees would destroy the usefulness of the method, but we next analyze the operations involved in selecting a column (tree) to enter the basis in a simplex algorithm for the transformed linear program.
The standard simplex criterion for a minimization problem seeks a column to enter the basis whose relative cost is most negative. If we associate the simplex multipliers $U_p$ with the capacity constraints, and $V_q$ with the convexity constraints, we have a row vector $(U, V)$ of multipliers:

$$(U, V) = (u_1, u_2, u_3, u_4, u_5, v^M, v^N)$$

for our example of two commodities on a five node network.

The implicit cost $z^{Mi}$ of a column $M^i$ is

$$z^{Mi} = (U, V) M^i = (U, V) (T^Mi, e^i)$$

$$= U^T M^i + V e^i$$

$$= U^T M^i + V = S(u_k T_k^Mi) + V^M$$

Similarly

$$z^{Ni} = U^T N^i + V = S(u_k T_k^Ni) + V^M$$

The relative cost $c^{-Mi}$ of a column (i.e., of routing plan $i$ for commodity $M$) is the difference between the true cost and the implicit cost: If the true cost $p^{Mi}$ less than the implicit cost $z^{Mi}$ it pays to enter the plan into the basis.

$$c^{-Mi} = p^{Mi} - z^{Mi}$$

and the relative cost $c^{-Ni}$ will be negative in that case.

Empirical studies have shown that an efficient computational rule is to find the column with the most negative relative cost: $\text{Min} \{c^{-Mi}, c^{-Nj}\}$. 
If all relative costs are positive for all commodities, the current basis is an optimal solution.

We therefore wish to find \( \min_{c} c_\text{Mi}, c_\text{Nj} \).

But now, by evaluating \( \min c_\text{Mi} \), we obtain a remarkable result for the decomposition algorithm as applied to multi-commodity flows.

We have that the true cost for a routing plan \( T_\text{Mi} \) is

\[
p_\text{Mi} = T_{\text{Mi}} = S) (t_k T_{k}^\text{Mi}),
\]

that is, the scalar product of the vector of true arc costs with the vector of flows on the arcs \( k \) in the tree \( T_\text{Mi} \).

The implicit cost is

\[
z_\text{Mi} = S) (u_k T_{k}^\text{Mi}) + V^M,
\]

this is the scalar product of the vector of implicit arc costs \( U \) with the vector of flows on the arcs \( k \) in the tree \( T_\text{Mi} \), plus the implicit cost attached to the convexity constraint for commodity \( M \).

Therefore, the minimum relative cost, for all plans \( i \) for commodity \( M \) is

\[
\min_{i} c_\text{Mi} = \min_{i} p_\text{Mi} - z_\text{Mi}
\]

\[
= \min_{i} S) (t_k T_{k}^\text{Mi}) - S) (u_k T_{k}^\text{Mi}) + V^M
\]

\[
= \min_{i} S) ( (t_k - u_k) T_{k}^\text{Mi}) + V^M,
\]

and similarly when \( N \) replaces \( M \).
Next notice the term $S) \left( (t_k - u_k) T_k^{Mi} \right)$. Define the quantities:

$$x_k = t_k - u_k,$$

for each arc $k$, $x_k$ is the difference between the true user cost $t_k$ and an implicit user cost $u_k$. We can define a row vector of relative arc costs $X$ as

$$X = T - U.$$

Now the term

$$S) \left( (t_k - u_k) T_k^{Mi} \right)$$

becomes

$$S) x_k T_k^{Mi} = XT^{Mi}$$

and the selection criterion

$$\min_i -c^{Mi}$$

becomes

$$\min_i XT^{Mi} - V^M.$$

Thus the selection criterion for commodity $M$ requires that we search through all trees $T^{Mi}$ of feasible flows to find that tree which yields the smallest cost $XT^{Mi}$, in terms of the relative arc costs $X$.

The selection criterion that the relative cost be negative states
We therefore seek, for some commodity, say $M$, a plan $T^M$ whose total travel cost in terms of the relative costs $X$, is less than the simplex multiplier $V^M$ attached to the convexity constraint for that commodity.

The form of the condition

$$\min_i X T^M_i = V^M$$

is the "remarkable result" to which we referred in an earlier paragraph.

It is remarkable because it states that in order to find a column to enter the basis, we must find a feasible flow for the subprogram corresponding to some commodity $M$. If we had written out all possible basic solutions to this subprogram, we would search among the columns (trees) of the constraint matrix, as in the usual simplex. However, we need not have written out all the columns in advance! This criterion, which asks for a minimum cost flow pattern, gives us a rule by which we can construct the column to enter the basis. This rule tells us to solve the minimum cost transshipment problem, given costs $X$ for some commodity $M$. The solution to this problem will be a tree flow pattern, that is, a vector $T^M_i$. If this tree has a cost less than $V^M$, we can augment the tree vector $T^M_i$ by the proper unit vector and form a new column to enter the basis, $M^i$. We also have the option of performing this calculation for every commodity.
and entering the column with the least relative cost among all commodity solutions. If no minimum cost commodity solution satisfies the criterion, the current solution is optimum.

The minimum cost transhipment problem can be solved by any of a number of algorithms written for that purpose. However, Pinnell observed that where there is only one destination in each commodity subprogram, and there are no capacity constraints particular to that subprogram, the solution to the minimum cost flow problem is simply given by a minimum path tree (in terms of relative costs $X_i$, of course) from each origin for that commodity to the destination. Minimum path trees can be constructed rapidly using, for example, Dantzig's algorithm.

We stated at the beginning of this section that the decomposition algorithm exploits the fact that for the revised simplex algorithm, one refers to a column of the constraint matrix only when selecting a column to enter the basis.

We have derived above a technique for constructing a column to enter the basis. It follows that in the decomposition algorithm we need never write out all columns in advance (this, in fact, is the main virtue of the algorithm) but, as in the revised simplex method, we need start with only enough columns to form an initial basis, and then we proceed to optimize by sequentially constructing the columns to enter the basis (via a minimum cost transhipment algorithm or a minimum path algorithm).

This completes our discussion of the computational aspects of the decomposition algorithm.

We next proceed to an analysis of the significance of the simplex multipliers in the multi-commodity flow problem.
The interpretation we shall present is based on one developed in terms of the theory of the firm by Fabian and Baumol.

4.5 Simplex Multipliers in the Multi-Commodity Flow Problem

In the transformed linear program, we have a simplex multiplier $u_k$ related to the capacity constraint on the k'th arc, and for all arcs, and a simplex multiplier related to the convexity constraint for the i'th commodity, for all commodities. The capacity constraints bound the solution from above, and the objective function, travel cost, is to be minimized. If we multiply the objective by $(-1)$ and maximize, we have an equivalent program. This program seeks to maximize profit (defined as the negative of the travel time loss) by choosing the optimal mix of basic solutions to the single commodity transhipment subprograms.

This formulation is completely analogous to the case discussed by Fabian and Baumol, in which a corporation, consisting of several divisions, desires to maximize profit. Each division has its own set of linear/technological constraints, and all divisions compete for scarce resources from the corporate pool. The divisional managers correspond, of course, to the players associated with the single commodity subprograms, in the traffic problem, and the resources in the corporate pool correspond to the lane capacities available to all commodity flows. The divisional technological constraints correspond to the conservation of flow and the input/output requirements for the commodity subprograms.

The solution to the corporate production planning problem is a set of production plans for the divisions, and their mixes. Each divisional plan is a basic solution to the sub-
program corresponding to that division. The initial basic solution is made up of any set of feasible divisional plans and their levels. That is, we construct a basis for the space of capacity and convexity constraints out of extreme points of the subprograms, and finding the levels which yield a feasible solution to the corporate program.

A simplex pivot step in the corporate program consists in soliciting from some divisions a proposal for a new divisional plan to enter the basis. In the traffic problem, this means seeking an alternative feasible flow pattern (tree) for some commodity. The proposed plan must be such that its relative profit is positive (in our above discussion, the equivalent criterion was a negative relative cost).

The following paragraphs interpret the calculation of the relative profit.

A proposed plan consists of a column vector of resource consumption coefficients per unit level of the activity, and a direct profit per unit level of the activity.

The column vector of consumption coefficients is augmented by a column unit vector which spans the convexity constraints, and has the unit in the row corresponding to the convexity constraint relative to the division submitting the plan.

In the traffic problem the resource consumption coefficients for a plan are the arc flows for a given tree, and the direct profit if the negative of the scalar product of the vector of arc flows with the vector of arc costs. The unit vector spanning the convexity constraints assures that any mixture of tree flows sum to a feasible solution for the commodity - the levels at which the various tree flows (or plans) are operated must sum to one.
A simplex multiplier $u_k$ is attached to each capacity constraint. This multiplier is the "shadow price", or value relative to the current basis, of the resource limited by this constraint. The shadow price is equal to the direct profit of the plan which is basic in the $k$'th row of the canonical form relative to the current basis.

The shadow price $u_k$ corresponds to the marginal revenue product of the $k$'th scarce resource. A unit increase in the activity basic to the $k$'th constraint, made possible by a relaxation of the constraint, permits an increase in profit to the system equal to the unit profit coefficient for that activity.

A simplex multiplier or shadow price $V^i$ is associated with the convexity constraint for each division of the corporation. An argument similar to that used for the $u_k$ shows that $V^i$ can be interpreted as the marginal revenue product of the $i$'th convexity constraint. If the right hand side of this constraint, which is unity, were doubled, the activity levels of all plans for $i$'th division could be doubled. Therefore the profit level of the $i$'th division would be doubled. The shadow price $V^i$ thus equals the current profit obtained from the operation of the $i$'th division under current mix of plans.

Formally, the analogy between divisions of the corporation and commodities on the network holds for the interpretation of the shadow prices $u_k$ and $V^i$. The shadow price $u_k$ is the marginal revenue product of the capacity on the $k$'th arc. An increase in capacity would permit an increase in flow on that arc. Some tree (or commodity flow plan) is basic to that arc, and the capacity increase (or constraint relaxation) which permits a unit increase in the level of that plan entails
an increase in the profit to the system equal to the unit profit coefficient for that plan. However, in the network problem, the profits (to be maximized) equal the negative of the travel costs for the various plans. Increasing the level of a plan therefore yields an increased cost. This seems paradoxical since providing more of a scarce resource results in a decreased profit (increased cost). This paradox is only due to the antisymmetry in the sign of the objective function between the corporate problem and the network problem. The paradox is easily resolved by observing that in both cases there is competition among divisions (commodities) for scarce resources. The shadow price of a scarce resource is interpreted as the unit profit (or negative cost) of the basic plan currently related to that scarcity constraint. If another activity were substituted in the basis, the shadow value would change to the unit profit of the new plan. In terms of flow plans on the network, one would substitute one unit cost value for another. The shadow price merely measures the current utility of a resource. The fact that one does not really wish to increase the level of that flow plan only (without for example reducing the level of a more costly flow plan), merely indicates the fact that the marginal revenue product analysis appears more "natural" when interpreted with respect to the maximization of a positive profit functional than with respect to the maximization of a negative profit functional (i.e., the minimization of positive costs).

The shadow prices $u_k$ on capacities may be thought of simply as representing the current utility of an arc's capacity.

The shadow prices $v^i$ represent, in the network problem, the total negative profit, (or cost) associated with the
flow of each commodity i.

We are finally ready to interpret the evaluation of a plan proposed to enter the basis. We are still considering the network problem as one of maximizing negative profits.

To each proposed divisional plan corresponds a set of resource input coefficients. A flow plan on the network corresponds to a vector of flows on arcs. This vector of flows equals the vector of arc capacities required to transmit those flows, hence the flows are the resource input coefficients.

A division is to be charged for the resources it proposes to consume. Resources are evaluated at their current shadow prices \( u_k \). The total shadow cost of resources to be consumed is deducted from the direct profit coefficient for that activity. This gives an adjusted profit. We have almost calculated the relative cost for this activity. We must, however, take into account the shadow price \( V_k \) associated with the convexity constraint. This becomes evident when we rewrite the formal condition for a plan to enter the basis. We form the relative profit \( c_{-i} \) by subtracting from the direct profit \( c_i \) for a plan, the implicit profit, where the latter is the scalar product of the activity associated input coefficient vector, \( V^i \), augmented by the convexity unit vector, with the vector of simplex multipliers \( u_k, V^i \). The relative profit \( c_{-i} \) therefore is

\[
  c_{-i} = c_i - \sum_k T^i_k u_k - V^i.
\]

For a plan to enter the basis, the relative profit must be positive so that we require

\[
  c_i - \sum_k T^i_k u_k - V^i \geq 0
\]
or, for the greatest marginal improvement,

\[ \max_k \left( c_i^i - \sum_k S_k T_k^i u_k \right) \geq v^i \]

This is clearly the negative of the criterion developed earlier in terms of positive costs and a minimization problem, as it should be. The term on the left is the net unit profit, after the implicit price of the resources consumed by a unit activity level of this plan has been subtracted from the direct unit profit. The criterion requires that the net unit profit exceed the current marginal profit of the current basic set of plans used by the division by which this plan has been submitted. This means that a marginal introduction of the proposed plan must yield a marginal net profit which is greater than the marginal profit to be gained if the current plans of the division were increased marginally. In other words, the proposed plan must be more profitable than "more of the same" old plans.

One can interpret these considerations for the network problem as follows.

A new routing plan is proposed for some commodity. It has a direct profit \( c_i^i \) (which is negative). It requires flow capacity as given by the vector \( T^i \). The arc capacities have current utilities given by the simplex multipliers \( u_k \) (they are negative). The value of capacities to be consumed by a unit level of this plan, given by \( S_k \) \( T_k^i u_k \), is subtracted from the direct profit for this plan \( c_i^i \) to form the net profit. This net profit is compared with the current marginal profit (negative) of the \( i \)'th commodity. As above, we have
After lengthy derivation, we are finally arriving at the sought-after interpretation. The introduction of an alternative routing plan for a commodity corresponds to the adoption of an alternative routing strategy by a player associated with the commodity. In the game theoretical model, each player choose his plan according to what ever information was available to him regarding the loading of the network. This information took the form of lists of true arc costs, and possibly information as to the saturation of certain arcs.

The true costs and saturation data were equivalent to information on the congestion costs (as in the case of a continuous congestion function). In the absence of cooperation among players each player chose the plan which appeared most profitable to him under the currently available cost information.

In the decomposition algorithm, each player is again asked to submit an alternative strategy, or plan. But, instead of considering only true travel times on non-saturated arcs, the player must balance the profit in terms of true travel times, $c_i$ for some plan $T_i$, against the implicit value $Z_i$ of the arc capacities utilized by this plan. The price $Z_i$ is subtracted from the direct profit $c_i$ for the plan, $T_i$, before this net profit can be compared with the current profit level for this commodity, $V_k$. The shadow prices on capacity, $u_k$, which generate the implicit value $Z_i$ by the formula

$$c_i - \sum_k T_i^k u_k = V_i$$

or, with $Z_i = \delta$ $T_i^k u_k$

$$c_i - Z_i = V_i$$
therefore play the roll of **tolls**, which the i'\text{th} player
must consider paying for the privilege of operating plan \( T^i \).

These are the tolls to which we referred in Chapter I in our discussion of the Pareto problem for the example of two parallel roads.

The net toll \( z^i \) for plan \( T^i \) is formed from the set of arc tolls \( u_k \). These are precisely the simplex multipliers, or shadow prices, on arc capacity.

These multipliers therefore carry the information required to force the individual players to select strategies which lead to a system optimum. Instead of evaluating routing plans in terms of direct profits (or true travel times) the player who is assumed to be following the rules of our decomposition "game" must evaluate strategies in terms of travel costs which are biased so as to lead him to choose plans which tend to improve the system cost.

How the toll mechanism operates will be illustrated in the following example. We shall apply the decomposition algorithm to the network used as a model for our game theoretical interpretation.

The algorithm begins with the construction of a basis for the associated linear program from a set of feasible solutions to the commodity subprograms.

The example showed that the plans \( T^{M2} \) and \( T^{N2} \) did not lead to a system optimal solution in the game theoretical model. Let us take these plans for our initial state in the decomposition program, and observe the progress of the algorithm to a system optimum.

\[
z^i = s_k \sum_{k} u_k
\]
The plans \( T^{N2} \) and \( T^{M2} \) were represented by the flow vectors
\[
T^{M2*} = (1,0,0,0,1) \\
T^{N2*} = (0,1,e,0,1).
\]

We augment these vectors by the proper convexity unit vectors to obtain the basic vectors (setting the perturbation \( e = 0 \)):
\[
M^{2*} = (1,0,0,0,1,1,0) \\
N^{2*} = (0,1,0,0,0,0,1).
\]

The basis is completed by the addition of slack vectors \( S_1, S_2 \ldots S_5 \):

\[
\begin{bmatrix}
S_1 & S_2 & S_3 & S_4 & S_5 & m^2 & n^2
\end{bmatrix}
\begin{bmatrix}
1 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0
\end{bmatrix}
\begin{bmatrix}
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1
\end{bmatrix}
= \begin{bmatrix}
H
\end{bmatrix}
\]

The first five columns \( S_1 \ldots S_5 \) correspond to the plan or activity of "storing" capacity on the five arcs. The levels at which these plans are executed are given by the variables \( S_1 \ldots S_5 \). The sixth column is the initial routing plan \( M^2 \) proposed for commodity \( M \); its level is \( m^2 \).
and its unit cost is \( 2 = P^{M2} \). The seventh column is the initial routing plan \( N^2 \) for commodity \( N \); its level is \( n^2 \) and the cost is \( 2 = P^{N2} \). Since there is only one current plan for each commodity, the convexity constraints require that the levels \( m^2 \) and \( n^2 \) both equal unity. The current flow pattern consists of a pure strategy for each player.

We continue with the decomposition algorithm by following the ordinary revised simplex procedure for the above basic tableau. If we name the \( i \)'th iteration basic matrix \( A_i \), and its cost row as \( P_i \), the vector of simplex multipliers \( W = (u_1, u_2, u_3, u_4, u_5, v^M, v^N) \) is given by \( W = P_i A_i^{-1} \).

We have

\[
A_1 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & -1 & 0
\end{bmatrix}
\]

\[
A_1^{-1} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]
\[ W = P A^{-1} = \]

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

\[ = (0,0,0,0,0,2,2) \]

or \[ u_1 = 0 \quad v^1 = 2 \]
\[ u_2 = 0 \quad v^2 = 2 \]
\[ u_3 = 0 \]
\[ u_4 = 0 \]
\[ u_5 = 0 \]

Naming the right hand side vector \( B \),

\[ B^* = (c_1, c_2, c_3, c_4, c_5, 1, 1) \]
\[ B^* = (1, 1, 1, 1, 1, 1, 1) \]

and naming the activity level vector \( X_1 \),

\[ X_1 = (S_1, S_2, S_3, S_4, S_5, m^2, n^2), \]

the current levels for the basic activities are given by
These activity levels check with the resultant net-
work flow obtained by plotting the flow patterns \( T^M_2 \) and \( T^N_2 \). \( T^M_2 \) saturates arcs 1 and 5, hence \( S_1 \) and \( S_5 \) are zero. \( T^N_2 \) saturates arc 2, hence \( S_2 \) is zero. There is no flow on arcs 3 and 4, so the slacks for these arcs are maximal: \( S_3 = S_4 = 1 \). Since each plan represents a pure strategy, \( m^2 = n^2 = 1 \). The simplex multipliers relative to the arcs are \( u_1 = u_2 = u_3 = u_4 = u_5 = 0 \). The shadow prices on the arcs are zero because, in the current basis, slack vari-
ables are basic relative to the capacity constraints.

The slack variables \( S_1 \), \( S_2 \) and \( S_5 \) are basic variables at zero level. Thus we have a degenerate solution. Fabian and Baumol point out that the simplex multipliers are not
meaningful in a degenerate solution. In the current solution, capacity on arcs 1, 2 and 5 is being used for a productive purpose. The related shadow prices should reflect the marginal utility of capacities on these arcs; they should not meaningfully be zero.

In order to remove the degeneracy, we perturb the right hand side. We increase each arc capacity by 10%, so that the new right hand side vector is

\[ B^\ast = (1.1, 1.1, 1.1, 1.1, 1.1, 1.1) \]

and the new solution is

\[ X^\ast = (A^{-1} B)^\ast = (0.1, 0.1, 1.1, 1.1, 0.1, 1.1) \]

or

\[ S_1 = 0.1, \quad S_2 = 0.1, \quad S_3 = 1.1, \quad S_4 = 1.1 \]

\[ S_5 = 0.1, \quad m^2 = 1, \quad n^2 = 1 \]

Under these circumstances, no arc is saturated since all slacks are positive. Therefore no capacity constraints are binding. All arc simplex multipliers are legitimately zero, since the marginal utility of increasing the supply of a resource which is not currently exhausted is zero.

Hence by perturbing the solution to eliminate degeneracy cost we revert to the usual interpretation of simplex multipliers. The fact that the arc tolls turn out to be zero in our example is due to the fact that slacks are needed to complete the basis. In a general problem, the slacks at
zero level will disappear when a sufficient number of alternative plans have been brought into the basis. Then resources used to capacity will have the proper non-zero multipliers attached to the related constraints.

The interpretation which we assigned to the convexity constraints does hold in this example. We have $V^M = 2$ and $V^N = 2$. These are the cost levels for the total flows of commodities M and N respectively. Under the current plan for commodity M (or N) were doubled, the cost would be double the value of $V^M$ (remember that this is strictly a marginal argument; the flow could not in fact be doubled without violating constraints).

Continuing the decomposition algorithm, we now request proposals from player M and player N for improved shipping plans. In the first part of this chapter, we showed that a proposed plan is a minimal cost shipping route plan given the adjusted costs on the network.

Later we showed that the adjusted costs are true costs less tolls, and the tolls are the simplex multipliers for the arcs.

The criterion for entering a plan into the basis was

$$\min_i \ x^T M_i \leq V^M$$

where $X$ is the vector of adjusted costs ($x_k = t_k - u_k$) and $T^M_i$ some basic solution to the commodity flow problem.

In the current stage of our example, the tolls are zero. Therefore we request from player M and player N respectively a tree flow pattern whose cost is less than the current commodity shipping cost $V^M$. 
Player M is already at his minimum cost plan, \( T^{M2} \), so he proposes no improving solution. Player N, however, proposes \( T^{N1} \) whose cost is 1, which is less than \( V^N = 2 \). Therefore \( T^{N1} \), augmented by the convexity unit vector, is a candidate for entering the basis we call. The augmented vector \( N^1 \), and its level, \( n^1 \).

Under these conditions of zero tolls the simplex criterion simply states that any commodity which has a shipping plan of lower cost than its current plan should enter this plan into the basis.

At this point we would continue the usual simplex technique to pivot \( N^1 \) into the basis. However, we run into another special situation at this point. Plan \( M^2 \) sends one unit of flow along arc 1; so would plan \( N^1 \). Clearly entering plan \( N^1 \) would drive plan \( M^2 \) out of the basis. But this would leave no plans for commodity M in the basis; this violates the convexity constraint. Alternatively, \( M^2 \) is driven to zero level and retained in the basis - introducing a degeneracy (and still violating the convexity constraint).

Under these circumstances we must pivot into the basis some other plan for commodity M, along with plan \( N^1 \) for commodity N.

If we choose plan \( M^1 \) for M, we should have an optimal solution, as we know from our direct analysis of this problem. We can construct this basis and check the simplex criterion for optimality.

The new tableau has the form
The new basis is

\[ A_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \]

With inverse

\[ A_2^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \]
and the new simplex multipliers are

\[ W = P^{-1} A_2 = \]

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

\[
= (0,0,0,0,0,2.5,1)
\]

Once again the arc tolls are zero because of degeneracy. The simplex criterion for the next stage requests player M and player N to propose a plan with routing costs less than 2.5 and 1, respectively. This time player N retains his current strategy, but player M proposes plan M'. In this case the simplex algorithm, like our game-theoretical interpretation, would cycle. Entering \( M' \) into the basis would drive \( N' \) out of the basis (or to a zero level within the basis). In order to avoid cycling, a perturbation scheme is resorted to, which guarantees finite convergence of the simplex algorithm (11).

4.6 Summary

The decomposition algorithm provides a method for modelling the interplay of commodity routing plans. The simplex algorithm guarantees convergence to a system opti-
mum in a finite number of iterations. The simplex multipliers on the arcs can be interpreted as tolls which lead to a system optimal choice of routes. In an example which is highly degenerate, the degeneracies can interfere with the interpretation of multipliers as tolls and with the convergence. Both these difficulties can be controlled through perturbation methods.

4.7 Priorities on the Network

In the convergent version of our game theoretical interpretation, players M and N acted independently and were in competition for the capacity of arc 1. (That is, for cheap transportation between nodes 1 and 2.) If player N was given priority in choosing his routing plan, he chose a plan containing arc 1, to the exclusion of player M. When player N had priority, the system cost was lower.

The linear programming technique yields a system optimal set of routing plans. From these plans, we can determine which commodities were allocated the high priority (that is, low cost) on arcs throughout the network.

In other words, the linear programming solution shows us which commodities ought to receive high priority in an optimal traffic planning scheme. In the following paragraphs we present some suggestions as to how the priority information obtained from a linear programming solution might be used to determine the flow control policy which leads toward an optimal flow pattern with the least cost of implementation.

This information can be applied in (at least) two ways. If we look at a given lane, the allocation of commodities to arcs tells us which commodities have high priority on that lane. Where a commodity is associated with an origin or des-
destination point, as in our example, the linear programming solution tells us the origin or destination of high priority flows on particular roads.

If, on the other hand, we look at a given commodity, we can determine the average priority for this commodity. Several approaches are possible. We may consider the entire network, and evaluate the (inverse) priority for a commodity by summing the arc costs. We may find the average priority per arc, for this commodity, and we may further find the average priority per vehicle. We may restrict priority calculations to subsets of the arcs in a network. Further investigation is required to determine which figure of merit may be best for a given example.

By applying these priority calculations, we can determine a rank order of commodity priorities on the network or on parts of the network. In studying practical traffic control problems, such an analysis would point out which traffic streams should receive the most attention if the network is to be used efficiently. Improvements to the network should be made so as to favor high priority flows. This conclusion provides an heuristic approach for traffic control and network improvement, based on a solution of the optimal flow problem.

Pinnell's work contains the germ of this idea, but not its generalization. In seeking better convergence for his decomposition algorithm (in the optimal flow problem), he found that certain commodities should be loaded on the network first. In his problem, only a small subset of arcs had capacity constraints. By loading the network with a commodity, and blocking these arcs Pinnell measured the damage
done to a commodity's flow plan by the blocking. Those commodities most affected were assigned high priorities. Assigning high priority commodities plans to the network first meant that if the plan belonged in the optimal basis, starting with the plan in the basis put the original basis closer to the optimal. This tended to reduce the number of pivots required to reach optimality. This is just a restatement of the general principle in linear programming that a good starting basis tends to reduce the number of pivots to optimality.

Our discussion is clearly a direct generalization of Pinnell's approach. Pinnell's particular method suggests a general technique for calculating priorities: Solve the minimal cost transshipment program for each commodity, and apply the parametric linear programming algorithm to study efficiently the effect of decreasing capacities on the flow cost for a commodity. This provides another measure of priority.

Finally, we suggest that the priority interpretation of the multi-commodity flow problem may be extendable to more general problems of queuing on a network. We may consider, for example, the modelling of a production line. The commodities may represent items arriving at input or output points at different times, hence we have a "dynamic network". An arrival may take various paths through the network, at various costs corresponding to capacitated arcs. A set of parallel arcs may represent a certain type of processing on machines of varying equality, efficiency, or speed. The model has the property that all items processed on parallel arcs (machines) leading into node i compete for processing on machines represented by arcs leading out of
node i. This implies that the difference in processing time among parallel machines is small. If this were not the case, dummy delay arcs could be added.

The problem is to determine priorities among different arrivals such that the output of the production line is maximized (or the cost minimized). We suggest that decomposition technique can be applied to this problem, which is similar to the vehicular traffic problem.
4.8 Notes to Chapter IV


(9) G. B. Dantzig, op. cit., Chapter 16.

5.1 Conclusions

In this thesis we have extended the wellknown multi-commodity linear programming model of traffic flow on a network.

We began with a description of the role of transportation models in locational analysis and rent theory. We constructed the "predictive" minimum path model with the "normative" minimum system cost model.

We next developed the linear programming "normative" model in detail, emphasizing that an excellent approximation to non-linear travel time function can be constructed using parallel "artificial arcs" to model lane behavior.

We then modified the usual multi-commodity network to permit the study of various road network synthesis problems. This was effected by introducing lane capacity as a variable in the linear program, subject to restrictions which depended on the particular synthesis problem.

In general, we required the capacity allocation activities to take on discrete values. This necessitated the solution of mixed-integer linear programs. We used a time-sharing computer system and the branch and bound algorithm to solve examples of these programs.

The first class of examples was based on the problem of finding the optimal pattern of one-way streets, given various network loading patterns.
We found that the solution was not intuitively obvious, and that a complete branch and bound algorithm was required to find all optimal and near optimal solutions. However, some shortcuts were discovered. When a branch and bound algorithm is relatively close to an optimum (i.e., good values of several integer variables have been chosen), other variables will assume values close to integers. This permits one to guess at a good integer solution, relatively early in the solution.

Secondly, it is possible to analyze the optimal flow pattern on the network in which capacity decisions are allowed continuous values. One then tries to improve the network (i.e., to capitalize on the non-linearity of costs by providing one-way streets) at the least possible disturbance to the current flow pattern. This means that one looks for lanes with very low flows in one direction and denies these lanes to those flows, for the benefit of flows in the opposite direction.

We also applied the branch and bound algorithm to a link addition problem. The method is appropriate, and more efficient than enumeration for problems which are not of trivial size.

We compared game theoretical interpretations of the multi-commodity flow problem with the decomposition interpretation. We interpreted the simplex multipliers on arc capacity constraints in the decomposition formulation as the tolls which lead to optimal utilization of the network. Degeneracy, a pathological situation in linear programming, can interfere with this interpretation. Finally, we presented a priority interpretation of the results of the linear programming analysis.
We have shown that the linear programming technique can be applied to determine lower bounds on vehicular travel costs in metropolitan traffic models, that it can be extended through mixed-integer programming to deal in a flexible manner with various problems of optimal network design, and that it can be interpreted to obtain information leading to optimal traffic control patterns.

5.2 Recommendations

A number of extensions of our model and solution technique should be investigated.

We solved our linear programs using a revised simplex code, but a code embodying the decomposition algorithm is more efficient, and capable of handling much larger problems. In applying the decomposition algorithm one may restrict the number of sources or sinks per commodity to one, and use a minimum path algorithm to find a vector to enter the basis. Alternatively, one may solve a general minimal cost transhipment problem to find a vector to enter the basis.

It has been found efficient in the ordinary revised simplex algorithm to compute more than one vector to enter the basis at a given iteration. After one is pivoted in, the relative cost for the second vector is checked, and if negative the vector is pivoted in. This technique is called "suboptimization".

In the multi-commodity decomposition approach, suboptimization can be effected by solving the minimum path or transhipment problem for several commodities. Rechecking the relative cost merely means reoptimizing the subprogram,
given a new set of multipliers. If we retain the previous
optimal flow solution to the subprogram, it should take
relatively little computational effort (a few primal pivots)
to revise the optimum. Hence suboptimization appears con-
venient in the multi-commodity program.

On the other hand, one subprogram differs from another
only in the input/output coefficients. The set of all sub-
programs in a given iteration corresponds to the solution
of one transhipment tableau for various input/output require-
ments. This suggests that one apply the dual simplex method
to transform a solution which is optimal for one subprogram
but infeasible for another, to one which is feasible for the
second.

One should investigate whether it is more advantageous
to gain space at the expense of time by keeping only one
transhipment tableau and obtaining the others by dual pivot
steps, or to gain time at the expense of spaces by keeping
the solutions for all tableaux (i.e., all commodity subprograms).

If one uses the minimum path tree approach, then these
considerations suggest an analogous investigation, which ap-
ppears of significant theoretical interest. In this approach
one calculates a tree of minimum length, rooted at one node.
The question is: is there an efficient way to pass from the
solution for one root node to that for another root node?

In our computations we have imbedded our linear program
within the branch and bound algorithm, to solve mixed-integer
programs. We conjecture that it is possible to solve the
mixed-integer program in the decompositions form by admitting
only extreme points of the capacity allocation subprogram,
which are trivial to calculate.
Ordinarily the algorithm will take convex combinations of extreme points to the capacity subprogram; the result will be non-integral values for decision variables. In the branch and bound algorithm, we obtain these non-integral values, and force them in succession to integral values. Each forcing requires a new solution of the flow problem. Clearly it would be much more efficient to solve the integer program within the decomposition program than vice-versa. But it must be proven that such a technique would in fact find the optimum.

There are other integer programming techniques which may be more appropriate. Land and Doig (1) gave a parametric version of their branch and bound algorithm which is probably more efficient than the version we used, although it requires more computer programming effort.

Our multi-commodity flow and network synthesis problem is a generalization of the "fixed charge transportation problem" of Hirsch and Dantzig (2). In this case there is a single commodity transportation problem without transhipment, and the fixed charge is assessed on flow across an arc, as in our link addition example. Balinski (3) presented an approximate method for solving this problem. Spielberg (4) adapted a partitioning method.

Whether these methods hold more, or less, efficiently for the multi-commodity transhipment problem remains to be investigated.

We turn now from questions of computational efficiency to extensions of the model. In the link addition model we proposed, all improvements were made in one time period. Extending the model to represent optimization over a number of
construction periods appears to be soluble by dynamic program-
ing. Naturally the flow input/output coefficients, or "de-
mands", must be expected to change through these periods.

This extension of the model would be a fairly complete representation of the long range road construction planning problem facing every local and state government.

Where the model represents rural flows, congestion effects are insignificant and the travel costs are linear with distance. In this case the minimal system cost solution is identical to the solution consisting of individual minimum paths, and the "normative" solution has the advantage of being also the predictive solution. The rural problem is being studied by Professor P. O. Roberts at M.I.T., using the nation of Colombia as a model.

The necessity of estimating demands brings up the field of stochastic programming. Our fixed-charge, multi-commodity problem remains to be investigated for stochastic demands.

Finally, the question of developing implementation techniques for the traffic control priority scheme developed in Chapter IV is a field of research which may yield significant contributions to the problem of reducing urban traffic congestion.
5.3 Notes to Chapter V


Appendix A: A Survey of Research on the Multi-Commodity Flow Problem

A.1 The Charnes-Cooper Multi-Copy Traffic Network Model

Let us begin with an elementary restatement of the Charnes-Cooper model (5, 6)*. We represent a traffic network by a linear graph. We define a flow $f_{ij}$ on this graph to be a function from the arcs to the non-negative reals satisfying

\[
S) \quad f_{ij} - f_{ki} = b_i \quad \text{for each node } i \quad (1)
\]

\[
i j \text{ in } L(i) \quad k i \text{ in } E(i)
\]

\[
f_{ij} \leq c_{ij} \quad (2)
\]

where

- $L(i)$ is the set of arcs leaving node $i$
- $E(i)$ is the set of arcs entering node $i$
- $c_{ij}$ is a function from the arcs to the non-negative reals, known as the arc capacity
- $b_i$ is the (positive or negative) input to the network of "flow" at node $i$, satisfying

\[
S) \quad b_i = 0 \quad i
\]

Defining $M$, the node-arc incidence matrix for the network, as having elements $m_{ij} = 1$ if $ij$ is in $L(i)$, $m_{ij} = -1$ if $ij$ is in $E(j)$, 0 otherwise, writing $\mathbf{f}$ as the vector of flows on arcs, writing $\mathbf{b}$ as the vector of inputs/outputs on nodes, and $\mathbf{c}$ the vector of capacities, conditions

* Numbers refer to items in Bibliography, Appendix B.
l and 2 become

\[ M \mathbf{f} = \mathbf{b} \] (conservation of flow) \hspace{1cm} (3)

\[ \mathbf{P} \preceq \mathbf{c} \] (capacity constraints) \hspace{1cm} (4)

We now introduce the notion of "kind" of flow, or "commodity". We may interpret this notion in a number of ways. For example we may consider the flow of vehicles entering the network at a given node (origin), desiring to have the network at a given node (destination) as one commodity. Or we may interpret the flow of vehicles entering the network at a given origin node, regardless of destination, as a commodity. This was the interpretation of Charnes and Cooper. One might further qualify the notion of kind of flow by distinguishing among vehicle types, cargo types, and priority categories.

Charnes-Cooper pointed out that, unless further qualification are introduced, the simultaneous flow of several commodities on one network is equivalent to the flow of each commodity on a separate replica, or copy, of the network. This is true because conditions (3) and (4) must hold independently for each commodity flow vector \( \mathbf{f}^k \) on the original network:

\[ M \mathbf{f}^k = \mathbf{b}^k \] (conservation of \( k \)'th commodity flow) \hspace{1cm} (3^1)

\[ M \mathbf{f}^k \preceq \mathbf{c}^k \] (capacity found for \( k \)'th commodity flow) \hspace{1cm} (4^1)

We next assume that there is a flow cost function, \( \tau_{ij}^k (f_{ij}^k) \), in units of travel time, giving the cost per unit of flow on arc \( k \) for commodity \( k \). We shall discuss the flow cost function in greater detail. For the moment it
suffices to point out that if the travel time is represented by a piecewise linear increasing function of flow, we can apply the methods of linear programming to solve the following problem:

Determine that set of commodity flows which satisfy (3') and (4') and produce a minimal total time for all commodities taken together. Note that in this case we solve k separate problems, and the total travel time is the sum of the travel times for the individual copies. Each problem is of the capacitated transhipment type, for which efficient computational algorithms exist (2). This problem is equivalent to the "classical" Hitchcock transportation problem.

On the other hand, if we stipulate in addition that the sum of all commodity flows on an arc ij is bounded by a common capacity limit \( c_{ij} \),

\[
\sum_{k} f_{ij}^k = c_{ij} \quad (5)
\]

then the different network copies are not independent; the flow on copy \( k \), is competing with the flow on other copies for a common scarce resource, capacity.
We remark that the separate capacity constraints $c^k$ could be dropped, and different incidence matrices $M^k$ for different flows introduced, without fundamentally altering the problem.

The multi-copy model is not equivalent to a classical transportation problem in the sense that here the constraint matrix is not unimodular. In the former case the unimodularity eliminates the need for division when pivoting from one basis to another, that permits the computational efficiency of the various transportation algorithms, and also insures integral flows when the inputs and outputs are integral. By contrast, the flows in the multi-copy problem are not necessarily integral. D. R. Fulkerson (4) presents a simple example. The multi-copy flow problems has also been termed the "Solid Transportation Problem" (33), the "Multi-Index
Problem" (32), the "Multi-Commodity Network Flow" problem, the "Group Assembly Problem" (14), and "Simultaneous Flows in a Network" (37).

Before we sketch the developments relating to the multi-copy problem from other contexts, we shall summarize the main assumptions of the Charnes-Cooper multi-copy model.

(a) We assume different types of flow on a network can be defined in such a way that we can specify the inputs and outputs of each flow type at each node. If a flow type is defined as the set of vehicles with a common origin, then we must know the transfer volumes, or the origin-destination desire lines, to use traffic planning parlance.

(b) We assume we can define a travel time function for each link which is a convex function of the loading of the link.

(c) We assume a flow configuration is desired which yields minimum total travel time on the network.

Assumption (a) seems unquestionably reasonable from a planning point of view; the establishment of desire line pattern has been the point of departure for the numerous metropolitan planning studies performed in this country. See (10) for a bibliography. Assuming transfer volumes can be specified does not relieve us of problems related to their statistical determination or the question of their rates of change in time. These questions are extensively discussed in the metropolitan planning literature. The Charnes-Cooper model takes these numbers as given.
Appendix A: Problems Related to the Multi-Copy Traffic Model

A.2 Multi-commodity feasibility

Ford and Fulkerson (19) presented the now famous "max-flow-min cut theorem" on the feasibility of a single commodity flow in a network: given a capacity function, a required input (called the flow value) at a single origin $s$ and an equal output at a single destination $t$, it is possible to construct a feasible flow from $s$ to $t$ over the network, i.e., a flow, which satisfies the continuity and capacity conditions and if, and only if the flow value is less than or equal to the capacity of a minimal cut separating $s$ and $t$. A cut is defined by partitioning the nodes of the network into two blocks $S$ and $T$, $W$ contains $s$ but not $t$, $T$ contains $t$ but not $s$. The cut is the set of arcs from $S$ to $T$, its capacity is the sum of the capacities of the arcs in the set. A minimal cut is the cut of least capacity, ranging over all partitions into $S$ and $T$. The theorem also holds if the term "cut" is replaced by "$s$, $t$ disconnecting set of arcs". Mayeda (47), Chien (9), and Gomory and Hu (26, 27), investigated the problem of "Multi-Terminal Flows": we are given a set of flow values from origin to destination $t$, for every $s$, $t$ pair in the network. Is each of these flows, taken separately, feasible on the network? Gomory and Hu showed efficient techniques for analyzing a given network for multi-terminal feasibility, and for synthesizing a network to meet flow requirements.

Note that "Multi-Terminal Flows" do not occur simultaneously on the network. S. L. Hakimi (30) attempted to generalize the max flow min cut theorem for simultaneous flows;
this case corresponds to the multi-commodity case. His conjecture on necessary and sufficient conditions was incorrect. Hu (30) published a counter-example and proved necessary and sufficient conditions for a two-commodity flow. Hu concluded that the only tool currently (1962-1963) available for constructing maximal multi-commodity flow was the Ford-Fulkerson linear programming technique (17) (which we shall discuss below). We would like to point out that the multi-copy model is appropriate for this purpose and that the application of a mixing (5) or decomposition (13) technique yields considerable computational efficiency. Hu further remarked that "since the necessity and sufficiency condition for the feasibility of multi-commodity flows is much more complicated than most people think, it is probably better to investigate multi-commodity flows in special networks such as telephone switching networks which are somewhat like bipartite graphs" (30). Nevertheless, Hu (38) later published a conjecture on the necessary and sufficient conditions for the feasibility of more than two simultaneous flows. Tang (62) derived a smaller set of conditions conjectured to be necessary and sufficient.

Tang (61) defined a class of bi-path networks for which he presented necessary and sufficient condition for multi-commodity flows. A network is bi-path if there do not exist three or more properly disjoint paths between any node pair of the network. A set of paths between two nodes properly disjoint if there is at least one intermediate node on each path and no intermediate node lies in more than one of the paths. Clearly this condition appears excessively restrictive with respect to road networks. Tang (60) presents a technique for synthesizing a minimal cost tree which satisfies multi-commodity feasibility requirements.
Hu's suggestion that the multi-commodity analysis problem (i.e., determination of feasibility given a network and set of flow requirements) and the multi-commodity synthesis problem (construct a minimum cost network to satisfy flow requirements) maybe more tractable in the case of special network structure appears to have been verified by Tang's results.

A.2.1 The solid transportation problem

The solid transportation problem represents a line of development of the multi-copy problem distinct from that arising from questions of flow feasibility. It is basically a generalization of the Hitchcock (36) classical transportation problem to more than two indices.

The Hitchcock problem can be stated as follows: given in sources and in links, find a flow $f$ on the arcs directly connecting the sources with the sinks, such that all supplies are distributed, all demands are satisfied, and the shipping cost summed over all routes is minimized. The shipping cost per route is the product of the constant cost $t_{ij}$ and the flow $f_{ij}$ on the route.

The problem is then

$$\min \sum_{S} f_{ij} t_{ij}$$
subject to \[ \sum_{j} f_{ij} = a_i \quad \text{supplies} \]

\[ \sum_{i} f_{ij} = b_j \quad \text{demands} \]

and \[ \sum_{i} a_i = \sum_{j} b_j \]

The problem is customarily represented in an \( m \times n \) tableau of the \( f_{ij} \) with row sums \( a_i \), column sums \( b_j \).

When intermediate nodes are introduced the problem is called the transhipment problem (53). The transhipment problem can be transformed to the Hitchcock problem when there are no cycles in the network around which the total cost is negative. (Dantzig (11)) In the multi-copy traffic flow problem, all travel costs are positive in the absence of subsidies, so that we can write the flow problem for each copy \( k \) as a transhipment problem and transform it to a transportation problem.

If we define \( f_{ij} \) as the amount flowing through node \( j \) we can write the flow conditions for one copy as

\[ \sum_{j}^{k} f_{ij}^{k} - \sum_{j}^{k} f_{jj}^{k} = b_j^{k} \]

(total flow \( k \) into node \( j \) minus flow through \( j \) vehicles leaving network at node \( j \))

\[ \sum_{i}^{k} f_{ij}^{k} - \sum_{i}^{k} f_{ii}^{k} = a_i^{k} \]

(total flow out of node \( i \) minus flow \( k \) through \( k \) vehicles entering network at node \( j \))
and
\[ \begin{align*}
S \) \( a^k_i &= S \) \( b^k_j = L; \\
\end{align*} \]

minimize
\[ \begin{align*}
S \) \( f^k_{ij} t^k_{ij} &= T^k \\
\end{align*} \]

Now define transhipment slack as
\[ f^k_{jj} = L - f^k_{jj} \]

Then the problem becomes
\[ \begin{align*}
S \) \( f^k_{ij} + f^k_{jj} = b^k_j + L & \quad \text{row conditions} \\
S \) \( f^k_{ji} + f^k_{ii} = a^k_i + L & \quad \text{column conditions} \\
\min S \) \( f^k_{ij} t^k_{ij} &= T^k \\
S \) \( a^k_i + L &= S \) \( b^k_j + L \\
\end{align*} \]

and it clearly has the form of the Hitchcock problem. Thus each copy of the multi-copy traffic network can be represented as a transportation tableau.

But it is necessary to represent the conditions which limit total flow on an arc, i.e., conditions of the form
\[ \begin{align*}
S \) \( f^k_{ij} \leq c_{ij} \\
\end{align*} \]

We might hope that the problem still remains equivalent to a transportation tableau, so we attempt to construct a
"supertableau", built from the row and column conditions for all copies. For a three by three tableau with two copies this becomes

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where the loops represent the flows tied together by common capacity constraints.

By constructing "two-copies in one tableau" we have presented the multi-copy model as a case of the transhipment problem with bounded partial sums of variables. The latter notion is credited to A. S. Manne by Dantzig (11), where the following theorem is stated: If a bounded partial sum
of variables includes variables in different columns or rows, the basis need not be triangular hence the problem is not equivalent to the transportation problem. This is simply another way of displaying the fact that the multi-copy constraint matrix is not unimodular.

Having failed to construct a transportation tableau by intermingling rows and columns of the different copies, we instead stack the tableau with the k'th copy in the k'th level in the stack. The common capacity bounds now correspond to vertical conditions: Fixing horizontal coordinates c, j, and requiring the sum over the vertical coordinate k to be less than $c_{ij}$, specifies the common capacity constraints.

We thus are led to a representation of the multi-copy traffic model as a three-dimensional transportation problem with equality constraints in two dimensions and inequality constraints in the third. By adding slack variables we can transform the inequalities into equalities. This is the "solid transportation problem". T. S. Motzkin (52) first generalized the Hitchcock problem to k indices. Schell (57) was interested in the case $k = 3$; he was interested in the Hitchcock problem of direct factory to warehouse shipments, with constraints on product types (flow type k) involving various sets of indices.

The method of solution was based on an extension of a primal "transportation" type algorithm.

Dwyer (14) cites Roby (56) as considering the generalized "assignment problem". It is well known that the two dimensional assignment problem (assign one man to each job so as to maximize the sum of their ratings r) is equivalent to a two dimensional transportation problem. Find numbers $f_{ij} = 0$
or 1 such that \( S) f_{ij} = 1 \) and \( \max_j S) f_{ij} f_{ij} \),

where \( f_{ij} = 1 \) if man \( i \) is assigned to job \( j \). In order to generalize this, Roby and Dwyer make the indices \( i, j, k \) ... correspond to job types; the value of the \( n \)'th index designates the number of the man assigned to the \( n \)'th job.

A rating factor \( G_{ijk} \) measures the relative efficiency of any group assignment of men to jobs. \( X_{i,j,k} = 1 \) if men \( i_k, j_k, k_k \) are assigned to the first, second, and third jobs. So \( X_{ijk} \) defines the set of possible group assignments. Constraints are imposed to assure the filling of all jobs and to prevent a man from being assigned to more than one job.

For three subscripts we are back to a solid transportation problem (or a multi-copy traffic network).

Galler and Dwyer (22) transformed the group and assembly problem to an equivalent minimal cost solid (or hyper) transportation problem. They generalized the "Hungarian method" (45, 16), or the two-dimensional assignment problem, to the \( n \)-dimensional case, calling the technique the "Method of Reduced Matrices". The Hungarian method, or "primal-dual algorithm" maintains dual feasibility (in the linear programming sense) and builds a primal (assignment) solution toward feasibility and optimality. In the reduced matrix method negative assignment variables may appear at "optimum". Galler trades off negativity for non-integrality of variables, and then obtains integrality by a procedure reminiscent of the artificial variable technique for the Simplex Phase I: adding a parametric variable which appears in those constraints which have fractional ("infeasible") values, one backs away from the optimum to some integral solution. It is not clear from the article whether one obtains the integer solution nearest to the fractional optimum. The problem of finding the integer solution
nearest to the fractional optimum is the essence of integer programming: that is the point of departure for the algorithm of Gomory (25), and Land and Doig (46). Galler (23) programmed for the IBM 709 an approximate solution technique obtained from an incomplete application of the reduced matrix problem.

In 1958, W. S. Jewell (40) published a report in which he too generalized the primal-dual transportation algorithm to the case of multi-commodity or multi-copy networks. He therefore was essentially applying the same technique to the same problem as that with which Dwyer and Galler were occupied at about the same time. However, by expressing the problem in terms of linked node-arc incidence matrices, rather than in terms of stacked transportation tableaux, and by considering the dual, he arrived at the following analysis for constructing minimal multi-commodity flow patterns.

A primal-dual solution consists in defining a restricted feasible network for each copy, and maximizing the flow in that copy by a simple flow maximizing routine, the Ford-Fulkerson labelling algorithm (19). Usually the first flow constructed on the restricted primal of a network copy does not satisfy the input-output requirements. A method is given for redefining the restricted primal in such a way that at the next iteration it will be possible to augment the flow. So far the algorithm is the same as for a single copy network. Jewell then reasoned that as long as flow values were low, i.e., at early stages in each copy problem, the flows were essentially independent. That means that the flows, when summed on capacitated links, would be below the common capacity bounds. Jewell then constructs an increasing sequence of dual-feasible, flows on the single copies. When the flows are large enough to begin to compete for common capacity, the dual variables, interpreted as potentials, direct the shifting of common capacity from one copy to another. An allocation of common arc capac-
ity to a network copy permits a flow augmentation for that copy at the expense of flow in that arc for another copy. The algorithm therefore consists of a direct flow phase (building flow in each copy) and an exchange flow phase (reducing one copy's flow in a common arc, increasing another's). The dual variables indicate the exchanges which will lead to an overall minimal cost flow pattern.

In Chapter IV we discuss computational experiments with Charnes's mixing routine for multi-copy networks in which primal feasible flows are mixed to provide an optimal solution, we shall see that Jewell's primal-dual algorithm provides a valuable hint toward computational efficiency in a primal method. Unfortunately, Pinnell (34), who developed the computer program for the mixing method, was apparently unfamiliar with Jewell's report, and arrived at a technique fundamentally similar to Jewell's (although a primal technique), only after long computational experience.

Returning to the stacked tableau form of the multi-commodity problem, we point out that Haley (32, 33) has formalized the work of Shell, and justified the extension of a primal transportation algorithm to the n-dimensional case. Haley calls the method the "modi-method", the English terminology appears to refer to Charnes's and Cooper's (7) "Stepping Stone Method".

Finally, K. Maghout et al (49) present a multi-copy model in which two transportation problems are linked through a common variable. The model arises in finding the optimal routing of milk from farms to plants, and from plants to markets, with total plant capacity greater than total market demand (to provide for fluctuations in demand), the slack
plant capacity ties together the two transportation problems. The dual of the problem is solved by the "Stepping Stone Method". The dual matrix has a "multi-stage" structure and could also be solved by a mixing or decomposition routine.

This example shows that the multi-copy model may be thought of quite generally as any linked set of transportation or transhipment problems - the copies need not refer to simultaneous flows on one network.

A.2.2 The arc-chain formulation for multi-commodity flows

Our final alternative formulation of the multi-commodity flow problem is due to Ford and Fulkerson (17). Instead of using a node-arc incidence matrix, an arc or commodity chain incidence matrix is defined as follows. A commodity chain is a set of arcs forming a path from an origin to a destination over which some flow "type" passes. Set up a chain vector $E$ with a component for each arc $i$ in the network; the component has the value 1 if arc $i$ is in chain $j$, 0 otherwise. Imagine now that we can list all the commodity chains in the network. The collection of vectors forms a matrix with a row for each arc. If we define the vector chain flow variables $f$, the scalar product of $f$ with a row of the matrix gives the total flow on an arc. This is constrained by the arc capacity $C_i$. To form a linear program for maximizing multi-commodity flow in the network, we take the arc-chain matrix, a right hand side of arc capacities, and a unit profit for each chain flow variable. We desire to maximize this objective function.
The power of the formulation lies in the observation that while the entire arc chain matrix would have an immense number of columns, to execute the revised simplex method we need only a basis and a column to enter the basis. The column to enter the basis is determined by the condition that the implicit profit of the column must be less than its direct profit. If \( a_{ij} \) are the elements of a column \( C_j \) and \( p_i \) are the implicit profits the condition is

\[
\sum p_i a_{ij} \leq 1.
\]

But this means that we want to find a path from some origin to some destination whose implicit profit is less than \( c \). We assign the implicit profits to arcs in the network, and search for a minimum path, from some origin to some destination. If the implicit profit, or length of this path (or chain) is less than 1, we have constructed a column (commodity chain) vector to enter the basis, and continue with the revised simplex method. If not we must try a minimum path for another origin destination pair. The practical advantage of this algorithm lies in the fact that it is relatively easy to compute minimum paths in a network via the Moore (51) or Dantzig algorithms (11). The great theoretical advantage lies in the fact that we have a subroutine for calculating the coefficients of a column to bring it into the basis. This eliminates the need to construct a priori the entire coefficient matrix. This notion suggested up a whole new class of linear programs (11), that of "programs with variable coefficients". This idea is fundamental to the decomposition algorithm, as is the mixing of solutions to subproblems, implicit in Jewell's primal-dual multi-commodity model.
Dantzig and Johnson (12) apply this Ford-Fulkerson algorithm in to find route of greatest payload per unit time and also the steady state flow in an air network, modified for greater efficiency by using the shortest path routine to induce an ordering on the arcs which permits deletions of arcs and rapidly reduces the network.

To conclude this survey we point out that Kalaba and Juncosa in 1956 (41, 42) suggested applying the multi-commodity flow model to finding optimal interoffice trunking patterns, to finding maximal flow patterns for messages in communications networks, and to synthesizing optimal improvements in communications networks. The latter problem was expressed as finding the minimum improvement necessary to improve system performance by a required amount.
Appendix B: Bibliography


BIography

Alan M. Hershdorfer was born in Newark, New Jersey on August 21, 1936. He attended public schools in that city, graduating from Weequahic High School in 1954. He attended Princeton University, majoring in physics. He was awarded the A. C. Williams Scholarship for 1954-1955, and the Union Carbide Scholarship, 1955-58. The academic year 1956-57 was spent at the University of Paris. He received the A.B. degree in June 1958.

He was a student in the Department of Physics at M.I.T. from September 1958 to September 1960, when he received the M.S. in Physics degree. The title of his M.S. thesis was, "A Data Reduction System for the M.I.T. Gamma Ray Satellite". During 1958-1959 he held an I.B.M. Research Assistantship at the M.I.T. Computation Center, and during 1959-1960 he was a research assistant with the Cosmic Ray Group of the M.I.T. Laboratory for Nuclear Science.

In 1960 he entered the M.I.T. Department of City and Regional Planning as a doctoral candidate. During 1960-1963 he held a National Defense Education Act Fellowship. During 1963-1964 he held a Samuel A. Stouffer Research Fellowship of the Harvard - M.I.T. Joint Center for Urban Studies, and was a research assistant in the M.I.T. Department of Civil Engineering. Since June 1964 he has been an instructor in that department.

On March 31, 1962 he married the former Selma Abigadol of Istanbul.